Асинхронное программирование с async await.

№ урока: 6 **Курс:** C# Асинхронное программирование

Средства обучения: Компьютер с установленной Visual Studio

Обзор, цель и назначение урока

Урок познакомит слушателей лекции с принципами асинхронного программирования. На уроке будут рассмотрены как разновидности асинхронных операций, так и асинхронность в целом. После этого урока студенты получат полное представление о понятии «асинхронность». Будут рассмотрены основные асинхронные шаблоны программирования, а также чистые асинхронные API для упрощения работы с асинхронным шаблоном TAP и ключевыми словами async await. Студенты научатся переписывать API старых асинхронных шаблонов согласно новому TAP. Для этого, на уроке рассматривается тип TaskCompletionSource. В конце урока будут рассмотрены ограничения в использовании оператора await.

Изучив материал данного занятия, учащийся сможет:

- Классифицировать асинхронные операции.
- Понимать асинхронные операции потоков.
- Понимать асинхронные операции ввода-вывода.
- Различать и знать принципы асинхронных шаблонов программирования.
- Переписывать API старых асинхронных шаблонов согласно новому TAP.
- Создавать свои асинхронные операции.
- Понимать, когда нельзя использовать модификатор async и оператор await.

Содержание урока

1. Асинхронные операции
2. Асинхронные CPU операции
3. Асинхронные операции ввода-вывода
4. Асинхронность
5. Асинхронные шаблоны программирования
6. Создание асинхронных операций
7. Ограничения в использовании ключевых слов async await

Резюме

- Асинхронная операция применяется как при ограниченной производительности ввода-вывода, так и при ограниченных ресурсах процессора, но по-разному в каждом случае.
- Асинхронные CPU операции применяются в случаях необходимости параллельного, неблокирующего или фонового выполнения, а также для распараллеливания операции.
- Асинхронные операции ввода-вывода используются при работе с файловой системой, сетью, базой данных и удаленными веб-сервисами.
- CPU операция – операция, выполняемая ресурсами (потоками) центрального процессора.
- Для создания асинхронной CPU операции воспользуйтесь статическим методом Task.Run().
- Асинхронные CPU операции используют многопоточность на для своего выполнения. Это означает, что они зависимы от ресурсов центрального процессора.
- Операции ввода-вывода – это операции передачи/получения сигнала (данных) между приложением/потоком и аппаратным обеспечением.

CyberBionic systematics

CyberBionic Systematics ® 2019
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

- Для работы с операциями ввода-вывода используют потоки данных. В .NET они называются стримами (Streams). К примеру, поток данных для работы с файловой системой – FileStream.
- Асинхронные операции ввода-вывода – это форма неблокирующей обработки операций ввода-вывода, которая позволяет потоку продолжить свое выполнение, не дожидаясь окончания передачи данных.
- Перекрывающий ввод-вывод (Overlapped IO) – название асинхронного ввода-вывода на уровне API операционной системы Windows. Представляется структурой OVERLAPPED.
- Завершение асинхронной операции ввода-вывода обеспечивается несколькими способами:
    - Событие Win32
    - Очередь APC (Asynchronous Procedure Call)
    - Порты завершения ввода-вывода (IO Completion Ports)
- Порты завершения ввода-вывода – это наиболее эффективное средство завершения асинхронной операции ввода-вывода.
- Порт завершения ввода-вывода – это объект, являющейся очередью, который используется для одновременного управления несколькими операциями ввода-вывода. Управление производится с помощью привязки дескрипторов к порту завершения.
- Чтобы FileStream работал в асинхронном режиме, используйте значение True для параметра isAsync или константу FileOptions.Asynchronous.
- Для асинхронных сетевых запросов используется класс HttpClient. Он не требует настроек для асинхронных запросов.
- Асинхронность не означает многопоточность. В большинстве случаев, асинхронность подразумевает использование асинхронных операций ввода-вывода. .NET же представляет единый формат для использования асинхронных CPU операций и операций ввода-вывода.
- Асинхронность – это неблокирующее выполнение кода.
- APM – Asynchronous Programming Model. Первый асинхронный шаблон программирования, основанный на интерфейсе IAsyncResult и методах BeginXXX и EndXXX. Метод BeginXXX используется для запуска асинхронной операции, а EndXXX для ожидания завершения, а также для получения результата (если такой есть у конкретной асинхронной операции). Поддерживает «callback» методы для обработки результатов асинхронной операции или продолжения работы. **Считается устаревшим.**
- EAP – Event-based Asynchronous Pattern. Асинхронный шаблон программирования, основанный на событиях. Асинхронную операцию представляет метод с названием XXXAsync, а за уведомление о завершении и передаче данных операции отвечает событие с названием XXXCompleted. Пример: асинхронная операция ReadAsync, событие о завершении операции ReadCompleted. Результаты, ошибки и другие данные асинхронной операции передаются через данные (класс производный от EventArgs) события в обработчики события. **Считается устаревшим.**
- TAP – Task-based Asynchronous Pattern. Асинхронный шаблон программирования, основанный на задачах. Для работы с шаблоном TAP создают асинхронные методы, которые возвращают задачу. Асинхронные методы имеют суффикс Async или TaskAsync в названии. Для упрощения работы с шаблоном TAP используются ключевые слова async await. **Рекомендуется к использованию.**
- Преимущества использования шаблона TAP:
    - Простая инициализация и завершение асинхронной операции.
    - Удобный способ получения возвращаемого значения асинхронной операции.
    - Получение исключения, возникшего в асинхронной операции для его обработки.
    - Просмотр состояния асинхронной операции.
    - Продолжения задач (Task Continuations/ async await).
    - Планирование выполнения асинхронной операции.
    - Поддержка отмены выполнения (Необязательно).

CyberBionic Systematics ® 2019
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

    - Поддержка прогресса операции (Необязательно).
- Для превращения API старых асинхронных шаблонов программирования в новый TAP используется класс TaskCompletionSource. Если необходимо преобразовывать API шаблона APM в TAP, то можно воспользоваться методами FromAsync фабрики задач (TaskFactory).
- TaskCompletionSource<TResult> - занимается созданием асинхронных операций в виде задач. Создает задачи-марионетки, которые можно завершить в любой момент времени результатом, ошибкой или отменой выполнения.
- Task.Delay – статический метод для создания асинхронной задержки на указанное время. Необходимую задержку можно указать в целочисленном значении (в миллисекундах) или в промежутке времени (структура TimeSpan). Доступна перегрузка с отменой выполнения. Поддерживается ключевыми словами async await.
- Task.WhenAll – статический метод для ожидания завершения всех переданных в качестве параметров задач.
- Task.WhenAny – статический метод для ожидания первой завершенной задачи из всех переданных в качестве параметров задач.
- Модификатор async запрещено использовать везде, кроме определения методов, анонимных методов и лямбда выражений.
- Оператор await запрещено использовать в следующих ситуациях:
    - В теле синхронного метода, лямбда выражения, анонимного метода.
    - В блоке оператора lock.
    - В большинстве выражений запроса (LINQ).
    - В небезопасном (unsafe) контексте.
    - Запрещено создавать экземпляры ref struct типом в асинхронных методах.
    - В блоке catch (Начиная с версии языка C# 6.0 – разрешено использование).
    - В блоке finally (Начиная с версии языка C# 6.0 – разрешено использование).
    - В методе Main (Начиная с версии языка C# 7.1 – разрешено использование).

Закрепление материала

- На какие разновидности можно разделить асинхронные операции?
- В каких случаях применяются асинхронные CPU операции?
- В каких случаях применяются асинхронные операции ввода-вывода?
- Каким способом в .NET можно создать асинхронную CPU операцию, соответствующую асинхронному шаблону TAP?
- Что такое операции ввода-вывода?
- Что такое асинхронные операции ввода-вывода?
- Чем отличается перекрывающий (Overlapped) ввод-вывод от асинхронного?
- Асинхронность – это всегда тоже самое, что и многопоточность?
- Что такое асинхронность?
- Сколько существует асинхронных шаблонов программирования?
- Какой из асинхронных шаблонов программирования актуальный и рекомендуемый к использованию?
- Какие преимущества использования асинхронного шаблона TAP?
- Что делает асинхронный метод Task.Delay?
- Что делает асинхронный метод Task.WhenAll?
- Что делает асинхронный метод Task.WhenAny?
- Когда запрещено использовать модификатор async?
- Когда запрещено использовать оператор await?

CyberBionic systematics

CyberBionic Systematics ® 2019
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

Дополнительное задание

Задание
Создайте приложение по шаблону WPF Application. Переместите из элементов управления (ToolBox) на форму текстовое поле и кнопку. Создайте асинхронный обработчик события по нажатию на кнопку. Используя класс HttpClient загрузите из Интернета html код любой страницы. Выведите его в текстовое поле.

Самостоятельная деятельность учащегося

Задание 1
Выучите основные конструкции и понятия, рассмотренные на уроке.

Задание 2
Создайте приложение по шаблону Console Application. Создайте асинхронный метод WriteToFileAsync, который в асинхронном режиме производит запись в файл. Организуйте ввод сообщений с клавиатуры в консоли. Результат ввода данных пользователем должен быть записан в файл с помощью вашего асинхронного метода WriteToFileAsync.

Задание 3
Создайте приложение по шаблону WPF. Переместите из элементов управления (ToolBox) на форму два текстовых поля и три кнопки. Создайте асинхронный метод CreateObjectsAsync(int objectCount). Который в своем теле создает массив типа object с количеством элементов, равным параметру objectCount. Далее, с помощью цикла инициализируйте каждый элемент массива новым экземпляром класса object. При этом, на каждой итерации сделайте асинхронную задержку на 1 секунду. По завершении инициализации массива выведите в текстовое поле количество созданных объектов в массиве. Создайте три асинхронных обработчика события для каждой из кнопок. Каждый обработчик внутри себя неблокирующим образом вызывает метод CreateObjectsAsync. Первый обработчик передает значение 250, второй – 400, третий – 1000.

Создайте асинхронный метод, который каждые 500 миллисекунд будет замерять количество занятых байт на куче (GC.GetTotalMemory()) и выводить во второе текстовое поле. Запустите метод при старте приложения.

Задание 4
Создайте приложение по шаблону Console Application. Создайте асинхронный метод, который асинхронно загружает html код сайта itvdn.com. Создайте асинхронный метод, который асинхронно считает количество упоминаний аббревиатуры «ITVDN» скачанного html кода главной страницы. Выведите результат на экран консоли.

Задание 5
Создайте приложение по шаблону Console Application. Запросите у пользователя любое число. Создайте асинхронную операцию с помощью класса TaskCompletionSource, которая в контексте потока из пула, считает всю последовательность чисел от 0 до указанного пользователем числа. Результат задачи выведите на экран консоли.

Рекомендуемые ресурсы

MSDN: Async overview
https://docs.microsoft.com/en-us/dotnet/standard/async

MSDN: Async in depth
https://docs.microsoft.com/en-us/dotnet/standard/async-in-depth

CyberBionic systematics
CyberBionic Systematics ® 2019
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine
Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

MSDN: Asynchronous Programming
https://docs.microsoft.com/en-us/dotnet/csharp/async

MSDN: Asynchronous Programming Patterns
https://docs.microsoft.com/en-us/dotnet/standard/asynchronous-programming-patterns/

MSDN: TaskCompletionSource
https://docs.microsoft.com/en-us/dotnet/api/system.threading.tasks.taskcompletionsource-1?view=netframework-4.8

MSDN: Task.FromAsync
https://docs.microsoft.com/en-us/dotnet/api/system.threading.tasks.taskfactory.fromasync?view=netframework-4.8

MSDN: Task.Delay
https://docs.microsoft.com/en-us/dotnet/api/system.threading.tasks.task.delay?view=netframework-4.8

MSDN: Task.WhenAll
https://docs.microsoft.com/en-us/dotnet/api/system.threading.tasks.task.whenall?view=netframework-4.8

MSDN: Task.WhenAny
https://docs.microsoft.com/en-us/dotnet/api/system.threading.tasks.task.whenany?view=netframework-4.8

CyberBionic systematics

CyberBionic Systematics ® 2019
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com