Введение в асинхронное программирование

№ урока: 1 **Курс:** C# Асинхронное программирование

Средства обучения: Компьютер с установленной Visual Studio

Обзор, цель и назначение урока

Данный урок является введением в асинхронное программирование на языке C#. Для правильного понимания технологий и методов их применения вы ознакомитесь с основной терминологией и рассмотрите примеры, когда необходимо применять техники асинхронного программирования. На этом уроке вы сможете наглядно увидеть, как асинхронность в прямом смысле оживляет приложения различных шаблонов (Console, WPF).

Изучив материал данного занятия, учащийся сможет:

- Понимать основную терминологию, связанную с асинхронным программированием.
- Разбираться, в каких случаях необходимо применение техник асинхронного программирования.
- Использовать пул потоков (Thread Pool).

Содержание урока

1. Понятия синхронности и асинхронности
2. Разбор термина «асинхронное программирование»
3. Разбор термина «параллельное программирование»
4. Применение асинхронного программирования
5. Потоки
6. Пул потоков

Резюме

- Поток (Thread) – координируемая единица исполняемого кода. Своим происхождением этот термин обязан понятию «поток исполнения». При организации многозадачности на основе потоков у каждого процесса должен быть по крайней мере один поток, хотя их может быть и больше. Это означает, что в одной программе одновременно могут решаться две и более задач.
- Синхронность – выполнение метода в контексте текущего потока.
- Асинхронность – выполнение метода в контексте вторичного потока.
- Асинхронное программирование – подход к написанию кода, который позволяет выполнять второстепенные задачи, не блокируя основной поток выполнения.
- Параллельное программирование – физическое выполнение нескольких операций одновременно. Достигается путем аппаратных возможностей вычислительной техники, а именно благодаря наличию нескольких ядер.
- Использование асинхронного программирования вместе с параллельным происходит по той причине, что иногда асинхронного программирования бывает недостаточно. Некоторые алгоритмы можно распараллеливать для более эффективной работы и повышения быстродействия.
- Явные случаи применения асинхронности:
  - Применение в UI (user interface), позволяющее не блокировать поток;
  - Второстепенные задачи;

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

    - Одновременная обработка нескольких клиентов;
    - Запросы в базу данных;
    - Сетевые запросы;
    - Работа с файловой системой.
- Пул потоков (Thread Pool) – это коллекция потоков, которые могут использоваться для выполнения методов в фоновом режиме.

Закрепление материала

- Что такое поток (Thread)?
- Чем отличаются синхронность и асинхронность?
- Что такое асинхронное программирование?
- Что такое параллельное программирование?
- Как асинхронное программирование может взаимодействовать с параллельным?
- В каких случаях следует использовать асинхронное программирование?
- Что такое пул потоков (Thread Pool)?

Дополнительное задание

Создайте проект по шаблону "Console Application". Создайте метод с названием WriteChar, который принимает один параметр типа char с названием symbol. В методе необходимо создать цикл for размерностью в 160 итераций, который будет выводить на экран консоли значение параметра symbol с задержкой в 100 миллисекунд. Из метода Main, используя пул потоков, организуйте параллельный вывод на экран двух символов звездочки и знака восклицания.

Самостоятельная деятельность учащегося

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

Задание 2
Создайте проект по шаблону "Console Application". Создайте класс-обертку для работы с классом Thread. Класс-обертка, должен позволить выполнить экземпляр класса-делегата Action<object> в контексте потока, созданного классом Thread. Он должен быть наделен свойствами bool IsCompleted (для проверки на завершенность выполнения метода), bool IsSuccess (для проверки на успешность выполнения), bool IsFaulted (для проверки на провал выполнения) и Exception типа Exception (для получения исключения, которое произошло в контексте вторичного потока).

Реализуйте также методы Start и Wait. Метод Start будет запускать экземпляр класса-делегата Action<object> на выполнение в контексте потока Thread. Метод Wait будет усыплять поток, который его вызвал, пока класс-делегат Action<object> не завершит свою работу.

По завершению выполнения нужно присвоить свойству IsCompleted - true. Если выполнение произошло без ошибок, свойству IsSuccess присвоить true, в противном случае - свойству IsFaulted присвоить true и в свойство Exception записать исключение. После создания класса-обертки повторить первое задание только с использованием своего класса-обертки.

Задание 3
Создайте проект по шаблону "Console Application". Создайте класс-обертку, параметризированный указателем места заполнения типом TResult для работы с классом Thread. Класс-обертка должен позволить выполнить метод, сообщенный с экземпляром класса-делегата Func<object, TResult> в контексте потока, созданного классом Thread. Реализовать свойства: bool

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

IsCompleted, bool IsSuccess, bool IsFaulted, Exception типа Exception как в третьем задании. Добавить свойство TResult Result, которое будет отдавать результат выполнения класса-делегата Func<object, TResult> в контексте вторичного потока.

Если результат еще не готов, то нужно усыплять поток, который его запросил, пока результат не станет доступным. Если выполнение было выполнено с ошибкой, свойство Result должно вернуть ошибку, а не результат.

Проверьте работу вашего класса-обертки, создав в классе Program метод Calculate с возвращаемым значением типа int и входящим параметром типа int с названием sleepTime. В теле метода Calculate в цикле for, размерностью в 10 итераций, проинкрементируйте значение переменной итерации цикла. На каждой итерации вызвать метод Sleep, усыпляя поток на значение входящего параметра sleepTime.

Верните из метода Calculate результат сложения в цикле. Выполните метод Calculate в контексте вашего класса-обертки. Пока результат не готов - выводите из метода Main на экран консоли знаки восклицания. Воспользуйтесь свойством IsCompleted чтобы узнать готов ли результат. Когда результат будет готов - выведите его на экран консоли.

Рекомендуемые ресурсы

MSDN: Асинхронное программирование
https://docs.microsoft.com/ru-ru/dotnet/csharp/async

MSDN: Шаблоны Асинхронного программирования
https://docs.microsoft.com/ru-ru/dotnet/standard/asynchronous-programming-patterns/

MSDN: Параллельное программирование
https://docs.microsoft.com/ru-ru/dotnet/standard/parallel-programming/

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