Исключения в асинхронном коде. Скоординированная отмена. Блокировки.

№ урока: 7 **Курс:** C# Асинхронное программирование

Средства обучения: Компьютер с установленной Visual Studio

Обзор, цель и назначение урока

Урок познакомит вас со способами поимки и обработки исключений в контексте потоков и задачах. Мы рассмотрим особенности обработки исключений с помощью ключевых слов async await. На этом уроке, мы также коснемся исключений и их обработки в контексте дочерних и вложенных задач. После этого, мы познакомимся с отменой выполнения задачи, то есть с так называемой скоординированной отменой. Чтобы писать асинхронный код качественно, необходимо знать о блокировках. На этом уроке мы рассмотрим некоторые ситуации, которые могут к ним привести, а также рассмотрим решения этой проблемы. Рассмотрим решение с использованием ключевых слов async await и настройк работы оператора await в виде метода ConfigureAwait. В конце урока мы рассмотрим прогресс асинхронной операции. Иногда это может помочь сделать ваше приложение более привлекательным для пользователя.

Изучив материал данного занятия, учащийся сможет:

- Ловить и обрабатывать исключения в контексте потоков.
- Ловить и обрабатывать исключения в задачах.
- Использовать ключевые слова async await для обработки как обычного исключения, так и составного.
- Знать об отношении родительской задачи к исключениям в дочерних и вложенных задачах.
- Отменять выполнение задачи.
- Понимать, что такое блокировка (deadlock) и самоблокировка.
- Знать способы исправления кода для избегания как обычных блокировок, так и самоблокировок.
- Создавать асинхронные методы, поддерживающие прогресс асинхронной операции.

Содержание урока

1. Что такое асинхронные исключения
2. Как обрабатывать исключения в контексте потоков и задач
3. Как обрабатывать исключения с использованием ключевых слов async await
4. Скоординированная отмена
5. Блокировка
6. Прогресс асинхронной операции

Резюме

- Исключение – это ситуация, при которой продолжение выполнения кода, в соответствии с заложенным программистом алгоритмом, уже невозможно или не имеет более никакого смысла.
- Асинхронное исключение – это исключительная ситуация, которая возникает в любой момент в другом потоке и при этом оно не зависит от того, какую конкретно инструкцию выполняет наш поток.

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

- Если исключение происходит в контексте вторичного потока, то и основной поток, который выполняет приложение будет аварийно завершен исключением.
- Обработка исключения в контексте вторичного потока должна происходить в методе, который был запущен через этот вторичный поток.
- Задача собирает все исключения, которые в ней произошли в специальное составное исключение AggregateException. Оно доступно через свойства задачи с названием Exception.
- AggregateException – класс, который может принять несколько исключений и записать их в себя. Он содержит коллекцию исключений асинхронной операции.
- Чтобы поймать исключения из задач, необходимо либо опрашивать свойство Exception, либо использовать следующее:
    - Методы ожидания: Wait, WaitAny, WaitAll
    - Свойство Result
    - Оператор await
    - Асинхронные методы ожидания: WhenAny, WhenAll
- Для обработки исключений из задач, необходимо обратится к одному из вышеуказанных методов, свойств или оператору await в блоке try конструкции try-catch. Если вы этого не сделаете, то исключение будет проигнорировано. В блоке catch необходимо обработать исключение или передать его дальше по стеку вызовов.
- Для удобства, редактор Visual Studio в режиме отладки демонстрирует возникшие исключения в контексте задач. Но это никак не влияет на выполнение. Исключение все также игнорируется, если вы не следуете предыдущему пункту.
- Отмена – прерывание операции по нашему запросу.
- Скоординированная отмена – отмена, которая требует помимо команды на прерывание выполнения операции, подтверждения на месте операции, которая должна быть отменена.
- Для обеспечения отмены в .NET используют три типа:
    - CancellationTokenSource – источник токенов отмены, класс для управления отменами.
    - CancellationToken – токен, проверяет наличие приказа на отмену, а также подтверждает, что нужно совершить отмену.
    - OperationCanceledException – исключение, представляющие отмену в .NET.
- Когда вы используете отмену выполнения с задачами, то вам необходимо отдавать задаче при создании токен отмены. Таким образом, отмена будет поддерживаться внутренней инфраструктурой задач. Многие методы для создания задачи имеют перегрузку для получения экземпляра CancellationToken.
- Чтобы сделать отмену через некий интервал времени, можно воспользоваться методом CancelAfter класса CancellationTokenSource.
- Токены отмены можно связывать между собой. Для этого используется статический метод CreateLinkedTokenSource.
- После произведения отмены выполнения – экземпляр класса CancellationTokenSource приходит в негодность. Чтобы повторно сгенерировать отмену, необходимо пересоздать экземпляр этого класса и выдать всем, кому необходимо новый токен отмены, связанный с этим новым экземпляром.
- Флаг LazyCancellation перечисления TaskContinuationOptions запрещает видеть отмену выполнения до завершения выполнения предыдущей задачи. Это позволяет сохранить целостность последовательного запуска продолжений при возможной отмене, если это важно.
- Блокировка (deadlock) – это ситуация, когда несколько потоков ожидают ресурсы, занятые друг другом. Это не дает продолжать выполнение.
- Самоблокировка – это ситуация, когда поток ждет завершения выполнения самого себя. При этом никто не дает ему сигнал для продолжения работы.
- Самое эффективное решение для классической блокировки – это использование общего, независимого ресурса в качестве объектов блокировки.

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

- Чтобы избежать самоблокировки, используйте для ожидания завершения и получения результата асинхронной операции оператор await.
- Если вы пишите библиотечный код, который потенциально будет использоваться из приложения, где присутствует контекст синхронизации, возвращающий выполнение в исходный поток, то лучше использовать метод GetAwaiter с указанием параметра false при ожидании вашей задачи с помощью оператора await. Иначе, разработчик, который вызовет этот метод и применит к нему блокирующие методы ожидания (Wait, WaitAll, WaitAny) или свойство Result, получит самоблокировку.
- Самым правильным решением при написании библиотечного кода можно считать – возвращать задачу программисту, который использует вашу библиотеку и пускай он сам решает ожидать ее оператором await или нет.
- Прогресс асинхронной операции — это информирование пользователей о состоянии выполнения асинхронной операции.
- Для работы с прогрессом операции в .NET существует интерфейс IProgress<T>. Он содержит абстракцию одного метода с названием Report. Суть этого метода - обновлять элемент или передавать информацию о состоянии выполнения операции.
- Если вам не требуется создавать свой класс для реализации интерфейса IProgress, вы можете воспользоваться стандартным классом Progress<T>. Также, его можно использовать в качестве базового класса для описания своего, производного. Класс может работать как с обычным делегатом Action<T>, так и с событием, которое будет инициировано при вызове метода Report базового интерфейсного типа IProgress.

Закрепление материала

- Что такое исключение?
- Что такое асинхронное исключение?
- Необработанное исключение вторичного потока разрушит работу основного потока?
- Каким способом можно обработать исключение, возникшее во вторичном потоке?
- Что происходит, когда исключение возникает в задаче?
- Что из себя представляет класс AggregateException?
- Исключения в дочерних и вложенных задачах ведут себя одинаково? В чем их различия, если они есть?
- Как можно поймать исключение, возникшее в задаче?
- Как обрабатывать исключения, возникшие в задачах?
- Что такое скоординированная отмена?
- Какие типы обеспечивают скоординированную отмену?
- Какой тип представляет саму отмену?
- Что такое блокировка?
- Что такое самоблокировка?
- Как можно избежать блокировок и самоблокировок?
- Что такое прогресс асинхронной операции?
- Какими типами в .NET представлен прогресс асинхронной операции?
- Какими способами класс Progress может информировать о состоянии асинхронной операции?

Дополнительное задание

Задание
Создайте приложение по шаблону WPF Application. Переместите из элементов управления (ToolBox) на форму два текстовых поля, две кнопки и прогресс бар. Сделайте приложение, которое считает факториал числа, введенного пользователем в одно из текстовых полей. Второе текстовое поле показывает результат. Приложение должно поддерживать отмену выполнения и с помощью прогресс бара демонстрировать прогресс операции. При этом, вы должны

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

обеспечить повторный запуск приложения с полной работоспособностью вычислений, отмены и прогресса операции.

Примечание: если операция выполняется слишком быстро, для наглядности добавьте задержку между вычислениями следующего промежуточного значения факториала.

Самостоятельная деятельность учащегося

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

Задание 2
В папке с материалами находится папка с решением под названием «HomeTasks». Откройте папку и запустите решение. Откройте первый проект с названием «001_Task». Данный проект запускает вторичный поток, который спустя время выбрасывает исключение, которое разрушает работу приложения. Исправьте этот изъян с помощью обработки исключения. Любое другое решение не подходит.

Задание 3
В папке с материалами находится папка с решением под названием «HomeTasks». Откройте папку и запустите решение. Откройте первый проект с названием «002_Task». Данный проект запускает задачу, которая выполняется фоново в контексте вторичного потока. Она должна выводить на экран консоли сообщение «Задача работает во вторичном потоке» раз в 2 секунды. Спустя какое-то время работы она перестает это делать. При этом приложение работает в обычном режиме, не указывая на какие-то проблемы. Где-то в задаче выбрасывается необработанное исключение. Измените код, чтобы не упустить исключение перед завершением приложения, а также добавьте его обработку.

Задание 4
В папке с материалами находится папка с решением под названием «HomeTasks». Откройте папку и запустите решение. Откройте первый проект с названием «003_Task». Это WPF приложение, в котором происходит обращение к 10 веб-ресурсам с задержкой в 1 секунду. Иногда при выполнении этого приложения выбрасывается исключение, которое указывает, что превышен лимит на обработку запроса. Вызов асинхронной операции ожидается с помощью оператора await, который может выбросить это исключение в основной UI поток. Добавьте обработку исключения из асинхронной операции.

Задание 5
В папке с материалами находится папка с решением под названием «HomeTasks». Откройте папку и запустите решение. Откройте первый проект с названием «003_Task». Это WPF приложение, в котором происходит обращение к 10 веб-ресурсам с задержкой в 1 секунду. На форме присутствует кнопка отмены. Но в самом коде не описана логика по отмене выполнения асинхронной операции по обращению к веб-ресурсам. Добавьте возможность скоординированной отмены.

Задание 6
В папке с материалами находится папка с решением под названием «HomeTasks». Откройте папку и запустите решение. Откройте первый проект с названием «003_Task». Это WPF приложение, в котором происходит обращение к 10 веб-ресурсам с задержкой в 1 секунду. На форме присутствует прогресс бар. Добавьте возможность отображения процесса обращения к 10 веб-ресурсам, чтобы пользователь мог видеть, что происходит загрузка. При завершении загрузки каждого из ресурсов изменяйте состояние прогресс бара.

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

Рекомендуемые ресурсы

MSDN: Exception class
https://docs.microsoft.com/en-us/dotnet/api/system.exception?view=netframework-4.8

MSDN: try-catch
https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/try-catch

MSDN: AggregateException
https://docs.microsoft.com/en-us/dotnet/api/system.aggregateexception?view=netframework-4.8

MSDN: Handling exception by await operator
https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/try-catch#exceptions-in-async-methods

MSDN: CancellationTokenSource
https://docs.microsoft.com/en-us/dotnet/api/system.threading.cancellationtokensource?view=netframework-4.8

MSDN: CancellationToken
https://docs.microsoft.com/en-us/dotnet/api/system.threading.cancellationtoken?view=netframework-4.8

MSDN: OperationCanceledException
https://docs.microsoft.com/en-us/dotnet/api/system.operationcanceledexception?view=netframework-4.8

MSDN: TaskContinuationOptions flags
https://docs.microsoft.com/en-us/dotnet/api/system.threading.tasks.taskcontinuationoptions?view=netframework-4.8

MSDN: IProgress
https://docs.microsoft.com/en-us/dotnet/api/system.iprogress-1?view=netframework-4.8

MSDN: Progress
https://docs.microsoft.com/en-us/dotnet/api/system.progress-1?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