Потокобезопасные коллекции. Producer-Consumer Pattern. Класс Parallel.

№ урока: 8 **Курс:** C# Асинхронное программирование

Средства обучения: Компьютер с установленной Visual Studio

Обзор, цель и назначение урока

Урок познакомит вас с потокобезопасными коллекциями из пространства имен System.Collections.Concurrent. На этом уроке будут рассмотрены коллекции, работающие по шаблону Producer-Consumer, это: ConcurrentQueue, ConcurrentStack, ConcurrentBag. Так как эти коллекции реализуют собой шаблон Producer-Consumer – на уроке будет рассмотрен и этот паттерн. Для его обеспечения мы познакомимся с потокобезопасной оболочкой BlockingCollection. После этого, мы познакомимся с классом для параллельных вызовов и параллельных циклов – Parallel. Будут разобраны вопросы по настройке выполнения этого типа и по слежению за состоянием параллельных итераций. А также, будет рассмотрена обработка исключений из класса Parallel.

Изучив материал данного занятия, учащийся сможет:

- Работать с потокобезопасной коллекцией ConcurrentQueue.
- Работать с потокобезопасной коллекцией ConcurrentStack.
- Работать с потокобезопасной коллекцией ConcurrentBag.
- Понимать, что из себя представляет паттерн Producer-Consumer.
- Работать с потокобезопасной оболочкой BlockingCollection.
- Выполнять параллельный вызов с помощью класса Parallel.
- Работать с параллельными циклами For и ForEach.
- Ловить и обрабатывать исключения параллельного вызова и циклов.
- Знать про внутреннее устройство потокобезопасных коллекций.

Содержание урока

1. Что такое потокобезопасная коллекция
2. Когда необходимо применять потокобезопасные коллекции
3. Какие техники синхронизации доступа используют потокобезопасные коллекции
4. Потокобезопасная коллекция ConcurrentQueue
5. Потокобезопасная коллекция ConcurrentStack
6. Потокобезопасная коллекция ConcurrentBag
7. Шаблон Producer-Consumer
8. Потокобезопасная оболочка BlockingCollection
9. Класс Parallel
10. Внутреннее устройство потокобезопасных коллекций

Резюме

- **Потокобезопасная коллекция** – это объект, который содержит сгруппированные данные с поддержкой их перебора, изменения, добавления или удаления безопасно из нескольких потоков.

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

- **Монопольное блокирование** – это реализация процесса выполнения кода, когда к определенному участку кода имеет доступ только один поток.
- Для повышения эффективности, потокобезопасные коллекции используют сразу несколько техник синхронизации доступа:
  - Простые конструкции и объекты синхронизации доступа: lock, Monitor, SpinWait.
  - Атомарные инструкции: volatile, Interlocked.
  - Неблокирующие алгоритмы.
- Разновидности потокобезопасных коллекций:
  - ConcurrentQueue
  - ConcurrentStack
  - ConcurrentBag
  - ConcurrentDictionary
- **ConcurrentQueue** – потокобезопасная коллекция, работающая по принципу FIFO (First In First Out). Для добавления элементов используется метод Enqueue, для извлечения элементов – TryDequeue.
- **ConcurrentStack** – потокобезопасная коллекция, работающая по принципу LIFO (Last In First Out). Для добавления элементов используется метод Push, для извлечения элементов – TryPop.
- **ConcurrentBag** – неупорядоченная потокобезопасная коллекция. Для добавления элементов используется метод Add, для извлечения элементов – TryTake.
- Шаблон Producer-Consumer подходит для ситуаций, когда скорость получения (генерации) данных/задач отличается от скорости обработки данных/задач.
- **Producer** – это изготовитель (поставщик) данных (задач), который создает или поставляет данные (задачи) в структуру данных.
- **Consumer** – это потребитель, который берет данные (задачи) из структуры данных и выполняет над ними манипуляции (обрабатывает, выполняет, отправляет результаты…).
- Интерфейс IProducerConsumerCollection реализует шаблон Producer-Consumer с помощью потокобезопасных коллекций.
- Класс **BlockingCollection** – объект, который является оболочкой для потокобезопасных коллекций, реализующих интерфейс IProducerConsumerCollection.
- BlockingCollection имеет ряд преимуществ:
  - Одновременное добавление (Add) и удаление (Take) элементов из нескольких потоков.
  - Поддержка ограничения и блокировки. Блокирование операции Add или Take, когда коллекция заполнена или пустая.
  - Возможность отмены выполнения методов Add/TryAdd и Take/TryTake с помощью CancellationToken или тайм-аута.
- Класс BlockingCollection имеет две реализации перечислителя:
  - Обыкновенный GetEnumerator – он возвращает перечислителя со «снимком» коллекции элементов. Под снимком имеется в виду получение элементов на момент вызова метода.
  - GetConsumingEnumerable - возвращает перечислитель, который будет отдавать (удалять из коллекции) элементы (если они есть в коллекции) до тех пор, пока значение свойства IsCompleted не станет равным true. Если элементов в коллекции нет и значение свойства IsCompleted равно false - цикл блокируется до тех пор, пока не появится доступный элемент или до отмены CancellationToken.
- Класс **Parallel** – это класс, который упрощает параллельное выполнение кода. У него доступно 3 параллельных API:
  - Invoke – параллельное выполнение делегатов Action.
  - For – параллельный цикл for.
  - ForEach – параллельный цикл foreach.

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

- Класс **ParallelOptions** – позволяет настроить выполнение параллельных методов. Имеет всего 3 настройки: максимальный уровень параллелизма, токен отмены, планировщик задач.
- Класс **ParallelLoopState** – позволяет отдельным параллельным итерациям параллельных циклов взаимодействовать друг с другом.
- Исключение, возникшее в одном из делегатов, выполняемого методов Invoke, не прерывает работу других делегатов или потоков. Делегат, в котором произошло исключение, прервет свою работу. Все исключения, возникшее в делегатах, собираются и помещаются в исключение AggregateException. Оно выбрасывается через точку вызова метода Invoke. Для обработки такого исключения необходимо помещать вызов метода Invoke в тело блока try конструкции try-catch.
- Исключение в одной из итераций параллельного цикла приводит к полному прерыванию работы всего цикла. Все исключения, возникшие в параллельных итерациях, собираются и помещаются в исключение AggregateException. Оно выбрасывается через точку вызова метода For или ForEach. Для обработки такого исключения, необходимо помещать вызов параллельных циклов в тело блока try конструкции try-catch.
- Структура **ParallelLoopResult** – предоставляет возможность просмотреть статус выполнения параллельного цикла.
- Все данные ConcurrentBag хранит в однонаправленном связном списке. Он представлен внутренним классом ThreadLocalList, экземпляр которого создается для каждого нового потока, который добавляет элемент в коллекцию. Каждый экземпляр этого класса имеет ссылку на созданный другим потоком экземпляр ThreadLocalList. Это и создает однонаправленный связный список из элементов ThreadLocalList.
- ThreadLocalList содержит внутри себя двунаправленный связный список. Он представлен внутренним классом Node. Этот класс хранит добавленные в коллекцию элементы. Каждый экземпляр класса Node имеет ссылку на следующий и предыдущий элементы Node. Из-за этого набор таких элементов становится двунаправленным связным списком.
- ConcurrentQueue состоит из экземпляров класса Segment, в которых сохраняются значения элементов по принципу FIFO. Каждый экземпляр Segment может хранить ссылку на следующий Segment.
- ConcurrentStack состоит из экземпляров класса Node, в которых сохраняются значения элементов по принципу LIFO. Каждый экземпляр Node может хранить ссылку на следующий Node.

Закрепление материала

- Что такое потокобезопасная коллекция?
- Что такое монопольное блокирование?
- Какие техники синхронизации доступа используют потокобезопасные коллекции?
- Какие потокобезопасные коллекции находятся в пространстве имен System.Collections.Concurrent?
- Что такое ConcurrentQueue?
- Что такое ConcurrentStack?
- Что такое ConcurrentBag?
- Что из себя представляет шаблон Producer-Consumer?
- Что делает каждый из участников шаблона Producer-Consumer?
- Какой интерфейс обеспечивает шаблон Producer-Consumer?
- Как называется класс, который является потокобезопасной оболочкой для потокобезопасных коллекций шаблона Producer-Consumer?
- Какими преимуществами обладает потокобезопасная оболочка для потокобезопасных коллекций шаблона Producer-Consumer?

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

- Сколько перечислителей предоставляет потокобезопасная оболочка для потокобезопасных коллекций шаблона Producer-Consumer?
- Что дает класс Parallel?
- Какие методы есть у класса Parallel?
- Что делает класс ParallelOptions?
- Что делает класс ParallelLoopStaet?
- Как исключения влияют на работу метода Invoke?
- Как обрабатывать исключения из метода Invoke?
- Как исключения влияют на работу параллельных циклов?
- Как обрабатывать исключения из параллельных циклов?
- Что делает структура ParallelLoopResult?
- Каким образом хранит свои данные ConcurrentBag?
- Каким образом хранит свои данные ConcurrentQueue?
- Каким образом хранит свои данные ConcurrentStack?

Дополнительное задание

Задание
Создайте приложение по шаблону Console Application. Создайте массив целочисленных элементов, размерностью в 10 000 000. Проинициализируйте массив с помощью параллельного цикла For от 0 до максимального размера. Создайте потокобезопасную коллекцию на свое усмотрение. Используя параллельный цикл ForEach переберите элементы массива и добавляйте в потокобезопасную коллекцию только те элементы, которые являются степенью двойки. Выведите на экран консоли элементы из вашей потокобезопасной коллекции.

Самостоятельная деятельность учащегося

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

Задание 2
Создайте приложение по шаблону Console Application. Создайте следующий класс:

```
internal class Product
{
    public string Name { get; set; }
    public int Quantity { get; set; }
}
```

Создайте класс Shop. Внутри него создайте:
- Коллекцию для хранения элементов типа Product.
- Метод с названием MakeAnOrder, в теле которого должен создаваться новый экземпляр класса Product и добавлять в коллекцию.
- Метод с названием ProcessOrders, в теле которого вы должны изымать из коллекции продукты и выводить на экран консоли название продукта и сколько единиц было куплено.

В классе Program используя задачи создайте несколько покупателей, которые будут делать несколько заказов, а также создайте одного сотрудника, который будет обрабатывать заказы.

Задание 3
Выполните задание под номером 2. Переделайте пример, используя шаблон Producer-Consumer. Вам необходимо использовать потокобезопасную оболочку BlockingCollection. Метод

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

ProcessOrders должен работать пока работа с оболочкой не завершена. Когда покупатели завершат покупку своих товаров, они должны об этом указать.

Задание 4
Создайте приложение по шаблону Console Application. Используя параллельный цикл ForEach прочитайте содержимое файла. Файл находится в папке с материалами. Название файла «data.txt».

Рекомендуемые ресурсы

MSDN: Thread-Safe Collections
https://docs.microsoft.com/en-us/dotnet/standard/collections/thread-safe/

MSDN: System.Collections.Concurrent
https://docs.microsoft.com/en-us/dotnet/api/system.collections.concurrent?view=netframework-4.8

MSDN: ConcurrentQueue
https://docs.microsoft.com/en-us/dotnet/api/system.collections.concurrent.concurrentqueue-1?view=netframework-4.8

MSDN: ConcurrentStack
https://docs.microsoft.com/en-us/dotnet/api/system.collections.concurrent.concurrentstack-1?view=netframework-4.8

MSDN: ConcurrentBag
https://docs.microsoft.com/en-us/dotnet/api/system.collections.concurrent.concurrentbag-1?view=netframework-4.8

MSDN: IProducerConsumerCollection
https://docs.microsoft.com/en-us/dotnet/api/system.collections.concurrent.iproducerconsumercollection-1?view=netframework-4.8

MSDN: BlockingCollection
https://docs.microsoft.com/en-us/dotnet/api/system.collections.concurrent.blockingcollection-1?view=netframework-4.8

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

MSDN: Parallel Options
https://docs.microsoft.com/en-us/dotnet/api/system.threading.tasks.paralleloptions?view=netframework-4.8

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

MSDN: ParallelLoopResult
https://docs.microsoft.com/en-us/dotnet/api/system.threading.tasks.parallelloopresult?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

Developer's blog PFX team: Exiting from Parallel Loops Early
https://devblogs.microsoft.com/pfxteam/exiting-from-parallel-loops-early/

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