ConcurrentDictionary. PLINQ

№ урока: 9 **Курс:** C# Асинхронное программирование

Средства обучения: Компьютер с установленной Visual Studio

Обзор, цель и назначение урока

Урок познакомит вас с потокобезопасной коллекцией ConcurrentDictionary. Эта коллекция будет рассмотрена в полном объеме. Начиная от рассмотрения всех API по работе с элементами, заканчивая настройкой и технической реализацией коллекции. Во второй части урока будет рассмотрен язык параллельных запросов Parallel LINQ (PLINQ). Этот урок познакомит вас с тем, как устроен PLINQ, какие у него есть операторы, а также как их можно настраивать. Также, не обойдем стороной отмену выполнения параллельного запроса и обработку исключений, возникших в нем.

Изучив материал данного занятия, учащийся сможет:

- Работать с потокобезопасной коллекцией ConcurrentDictionary
- Знать про внутреннее устройство потокобезопасной коллекции ConcurrentDictionary
- Выполнять параллельные запросы с помощью PLINQ
- Знать о параллельных операторах, которые доступны только в PLINQ
- Отменять выполнение PLINQ
- Ловить и обрабатывать исключения PLINQ

Содержание урока

1. Потокобезопасная коллекция ConcurrentDictionary
2. Методы по работе с элементами в ConcurrentDictionary шаблона TryXXX
3. «Всегда-выполняемые» методы по работе с элементами в ConcurrentDictionary
4. Внутреннее устройство потокобезопасной коллекции ConcurrentDictionary
5. Настройка ConcurrentDictionary
6. Технология PLINQ
7. Внутреннее устройство PLINQ
8. Параллельные операторы
9. Операторы по настройке PLINQ
10. Обработка исключений из PLINQ

Резюме

- **ConcurrentDictionary** – это класс, который представляет собой потокобезопасную коллекцию, работающую по принципу ключ-значение.
- ConcurrentDictionary не является реализацией шаблона Producer Consumer. У него нет в реализации интерфейса IProducerConsumerColletion.
- Методы по работе с элементами в ConcurrentDictionary можно разделить на две группы:
  - Методы, написанные по шаблону TryXXX
  - «Всегда-выполняемые» методы
- Методы, написанные по шаблону TryXXX:
  - TryAdd – пытается добавить значение в коллекцию по ключу.
  - TryGetValue – пытается извлечь по указанному ключу значение и поместить его в out параметр.

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

    - TryRemove – пытается удалить элемент по указанному ключу. Перед удалением элемент помещается в out параметр.
    - TryUpdate – пытается обновить значение по указанному ключу. Для обновления значения необходимо знать старое значение, которые находится до операции обновления. Эта проверка необходима для подтверждения актуальности обновления.
- «Всегда-выполняемые» методы:
    - AddOrUpdate – добавляет значение в коллекцию, если по указанному ключу его там нет. Если по указанному ключу уже есть какое-то значение, то происходит его обновление.
    - GetOrAdd – извлекает значение элемента по указанному ключу. Если по указанному ключу элемента нет, то происходит его добавление.
- Хранилище ConcurrentDictionary состоит из так называемых bucket-ов. Каждый бакет представлен экземпляром класса Node.
- Класс **Node** – это вложенный закрытый класс, который представляет собой однонаправленный связный список. Он содержит внутри себя: ключ, значение, ссылку на следующий экземпляр класса Node и хэш-код ключа. Для доступа к данным используется монопольная блокировка с помощью класса Monitor. У ConcurrentDictionary есть специальный массив, который хранит объекты для блокировок доступа. В массиве содержится сразу несколько объектов. Если вы не настраиваете словарь ConcurrentDictionary, то количество объектов для блокировки будет равно количеству логических процессоров вашего компьютера, на котором развертывается приложение, использующее ConcurrentDictionary.
- **Parallel LINQ** (PLINQ) – параллельная реализация LINQ to Objects. Его работу обеспечивает статический класс под названием ParallelEnumerable. В нем находятся как всеми знакомые операторы LINQ, так и новые операторы, которые принадлежат только PLINQ. Чтобы у вас была возможность вызывать операторы PLINQ, вам необходимо преобразовать последовательность IEnumerable в класс ParallelQuery.
- Оператор AsParallel() – доступен для вызова на последовательности IEnumerable<T>. Он превращает эту последовательность в ParallelQuery<T>, тем самым начиная работу с PLINQ.
- Оператор **AsOrdered**() – указывает PLINQ, что необходимо сохранять порядок элементов исходной последовательности.
- Оператор **AsUnordered**() – отключает указания метода AsOrdered(). После него исходный порядок может быть искажен. Полезно, когда у вас очень большие запросы или, когда вы используете упорядоченный готовый запрос как основу для нового запроса.
- Оператор **AsSequential** – указывает, что вся последующая часть запроса должна выполняться последовательно. То есть, он выключает работу PLINQ.
- Оператор **ForAll**() – многопоточный метод, который позволяет без слияния элементов параллельно обработать результаты PLINQ запроса указанным вами делегатом.
- Оператор настройки **WithCancellation** – позволяет передавать токен отмены для отмены выполнения параллельного запроса.
- Оператор настройки **WithDegreeOfParallelism** – позволяет указать максимальное количество логических процессоров, который PLINQ может использовать для параллельной обработки. PLINQ в праве не использовать указанное вами количество, он может задействовать меньше, чем вы указали. Но, он не перейдет за указанный вами порог.
- Оператор настройки **WithExecutionMode** – позволяет указать режим выполнения PLINQ. PLINQ не всегда может выполняться параллельно. При некоторых комбинациях операторов PLINQ может посчитать, что последовательный режим более необходим. С помощью WithExecutionMode вы можете заставить PLINQ всегда выполняться параллельно.

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

- Оператор настройки **WithMergeOptions** – позволяет указать способ слияния элементов. Имеет три различные настройки:
  - NotBuffered – требование немедленного возвращения обработанного элемента из каждого потока сразу же после его создания.
  - AutoBuffered – требование собирать элементы в промежуточные буферы и время от времени их очищать, отдавая элементы потоку-потребителю PLINQ запроса.
  - FullyBuffered – требование собирать элементы в буфер и только после полного наполнения возвращать элементы потоку-потребителю PLINQ запроса.
- Исключение из параллельных потоков обработчиков приводит к полному завершению выполнения запроса, даже если он не выполнил все свои операторы. Все исключения, возникшие при выполнении запросов PLINQ, собираются и помещаются в AggregateException, который будет выброшен через участок кода, который инициировал PLINQ запрос. Инициатором может выступать цикл foreach, метод ForAll или методы для генерации коллекций на основе запроса. Чтобы выброшенное исключение не сломало работу вашего приложения, вызов инициатора запроса необходимо поместить в тело блока try, конструкции try-catch.
- Если для вас критично, что вызывающий поток блокируется на время выполнения PLINQ запроса, то вы можете обернуть выполнение параллельного запроса в класс Task. И, воспользовавшись ключевыми словами async await, неблокирующим образом дождаться завершения выполнения задачи, которая будет ждать завершения выполнения параллельного запроса.

Закрепление материала

- Что из себя представляет класс ConcurrentDictionary?
- Можно ли использовать ConcurrentDictionary в шаблоне Producer-Consumer?
- Какие разновидности методов по работе с элементами есть в ConcurrentDictionary?
- Как устроено внутреннее хранилище ConcurrentDictionary?
- Что такое Parallel LINQ (PLINQ)?
- Как необходимо распараллелить LINQ запрос?
- PLINQ сохраняет исходный порядок элементов?
- Как параллельный запрос PLINQ превратить в последовательный?
- PLINQ поддерживает отмену выполнения?
- PLINQ всегда выполняется параллельно?
- Можно ли влиять на работу слияния элементов в PLINQ?
- Как ловить и обрабатывать исключения в PLINQ?

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. Используя Parallel LINQ (PLINQ) выберите все элементы, которые являются степенью двойки. Выведите эти элементы на экран консоли.

Самостоятельная деятельность учащегося

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

Задание 2
Создайте приложение по шаблону Console Application. Создайте следующие классы:

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

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


internal class Customer
{
    public string Name { get; set; }
    public string Phone { get; set; }
    public string Address { get; set; }
}

internal class Order
{
    public Customer Customer { get; set; }
    public List<Product> Products { get; set; }
}
```

Создайте класс Shop. Внутри него создайте:

- Коллекцию ConcurrentDictionary, которая по имени покупателя будет хранить его заказы.
- Метод с названием MakeAnOrder, в теле которого должен создаваться новый экземпляр класса Product и добавляться в коллекцию. Если там такой продукт уже есть, необходимо изменить его количество.
- 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

Создайте приложение по шаблону Console Application. Создайте массив целочисленных элементов размерностью в 1000. Проинициализируйте массив с помощью параллельного цикла For от 0 до максимального размера. Используя Parallel LINQ (PLINQ) выберите все нечетные элементы, сохраняя исходный порядок последовательности. Выведите эти элементы на экран консоли.

Задание 4

Создайте приложение по шаблону Console Application. Используя Parallel LINQ (PLINQ) прочитайте из файла все слова, которые начинаются на букву «a-A». Выведите эти элементы на экран консоли. 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: ConcurrentDictionary

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

https://docs.microsoft.com/en-us/dotnet/api/system.collections.concurrent.concurrentdictionary-2?view=netframework-4.8

MSDN: PLINQ
https://docs.microsoft.com/en-us/dotnet/standard/parallel-programming/introduction-to-plinq

MSDN: Understanding Speedup in PLINQ
https://docs.microsoft.com/en-us/dotnet/standard/parallel-programming/understanding-speedup-in-plinq

MSDN: When PLINQ Chooses Sequential Mode
https://docs.microsoft.com/en-us/dotnet/standard/parallel-programming/understanding-speedup-in-plinq#when-plinq-chooses-sequential-mode
MSDN: Order Preservation in PLINQ
https://docs.microsoft.com/en-us/dotnet/standard/parallel-programming/order-preservation-in-plinq

MSDN: Merge Options in PLINQ
https://docs.microsoft.com/en-us/dotnet/standard/parallel-programming/merge-options-in-plinq

MSDN: How to handle Exceptions in a PLINQ Query
https://docs.microsoft.com/en-us/dotnet/standard/parallel-programming/how-to-handle-exceptions-in-a-plinq-query

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