Дәріс 13. Конкуренттік коллекциялар.

Дәрістің мақсаты: Студенттерде конкуренттік коллекциялар қызметі туралы түсінік қалыптастыру.

Дәрісті меңгеру нәтижесінде студенттер келесі қабілеттерге ие болады:

- Конкуренттік коллекциялар түрлерін ажырату;
- Конкуренттік коллекцияларды стандартты коллекциялардан ажырату.

.NET Framework ортасының 4.0 нұсқасына жаңа System.Collections.Concurrent аттар кеңістігі қосылды. Ол ағын қауіпсіз болып табылатын және қатар бағдарламалауға арнайы арналған коллекцияларды қамтиды. Бұл дегеніңіз олар екі немесе одан көп қатар орындалатын ағындар тарапынан коллекцияға бір мезгілде қол жеткізуге болатын көп ағынды бағдарламада қауіпсіз пайдаланылуы мүмкін дегенді білдіреді.

- BlockingCollection<T> - Интерфейсті бұғаттайтын іске асыру үшін қабықты ұсынады IProducerConsumerCollection<T>.
- ConcurrentBag<T> - Ақпарат бір ағымда шығарылған және тұтынылған жағдайда неғұрлым жарамды көрсетілетін IProducerConsumerCollection<T> интерфейсін реттеусіз іске асыруды қамтамасыз етеді.
- ConcurrentDictionary<TKey, TValue> - "Кілт-мән" жұптарын сақтайды, яғни параллель сөздікті іске асырады.
- ConcurrentQueue<T> - Параллель кезекті және интерфейстің тиісті нұсқасын іске асырады IProducerConsumerCollection<T>.
- ConcurrentStack<T> - Параллель стек пен интерфейстің тиісті нұсқасын іске асырады IproducerConsumerCollection<T>.

Көріп отырғаныңыздай, параллель коллекциялардың бірнеше сыныбында IProducerConsumerCollection интерфейсі іске асырылады. Бұл интерфейс System.Collections.Concurrent аттар кеңістігінде де анықталған. Ол IEnumerable, IEnumerable<T> және ICollection интерфейстерінің кеңеюі ретінде қызмет етеді. Бұдан басқа, онда "жеткізуші-тұтынушы" үлгісін қолдайтын әдістер айқындалған. TryAdd() әдісі элементті жиынға қосуға, ал TryTake() әдісі элементті жиыннан жоюға тырысады. Төменде екі әдістің де хабарландыру нысандары келтірілген.

bool TryAdd(T item)

bool TryTake(out T item)

TryAdd() әдісі егер жиынға item элементі қосылған болса, логикалық мәнді true береді. Ал TryTake() әдісі егер item элементі жиыннан жойылса, логикалық мәнді true береді. Егер TryAdd() әдісі сәтті орындалса, онда item элементінде нысан болады.

Параллель коллекциялар көбінесе тапсырмаларды ашу кітапханасымен (TPL) немесе PLINQ тілімен бірге қолданылады. Бұл коллекциялардың ерекше сипатына байланысты олардың барлық сыныптары одан әрі егжей-тегжейлі қарастырылмайды. Оның орнына нақты мысалдарда BlockingCollection<T> сыныбына қысқаша шолу беріледі. BlockingCollection<T> класын құру негіздерін меңгере отырып, сіз басқа қатар топтамаларды да аса қиындықсыз шеше аласыз.

BlockingCollection<T> сыныбында, мәні бойынша, бұғаттау кезегі іске асырылады.

Бұл элементті толтырылған кезде жиынға кірістірудің кез келген әрекетін, сондай-ақ бос кезде жиыннан элементті жою әрекетін автоматты түрде күтуді білдіреді. Бұл "жеткізуші-тұтынушы" үлгісін қолданумен байланысты жағдайлар үшін тамаша шешім. BlockingCollection<T> класында ICollection, IEnumerable, IEnumerable<T>, сондай-ақ IDisposable интерфейстері іске асырылады.

BlockingCollection<T> сыныбында мынадай конструкторлар анықталады.

public BlockingCollection()
public BlockingCollection(int *boundedCapacity)*
public BlockingCollection(IProducerConsumerCollection<T> *collection)*
public BlockingCollection(IProducerConsumerCollection<T> *collection,* int *boundedCapacity)*

Алғашқы екі құрастырушыда BlockingCollection<T> сыныбының қабығына ConcurrentQueue<T> типті объектінің данасы болып табылатын коллекция жасалады.
Ал басқа екі құрастырғышта BlockingCollection<T> типті коллекцияның негізіне алынуы тиіс коллекцияны көрсетуге болады. Егер boundedCapacity параметрі көрсетілсе, онда ол құрсауланар алдында жиынтық қамтуы тиіс нысандардың ең көп санын қамтуы тиіс. Егер boundedCapacity параметрі көрсетілмесе, онда жиын шектелмеген болып шығады.

public void Add(T item)
public T Take()

Add() әдісі шектеусіз жиын үшін шақырылғанда, ол item элементін жиынға қосады және содан кейін бағдарламаның шақырушы бөлігін басқаруды қайтарады. Ал Add() әдісі шектеулі коллекция үшін шақырылғанда, егер ол толтырылған болса, оған кіруге тыйым салады. Жиыннан бір немесе одан көп элемент жойылғаннан кейін, көрсетілген item элементі жиынға қосылып, осы әдістен қайтарылады. Take() әдісі жиыннан элементті жойып, бағдарламаның шақырушы бөлігін басқаруды қайтарады.

Add() және Take() әдістерін қолдана отырып, төменде келтірілген бағдарлама мысалында көрсетілгендей, "тұтынушы жеткізуші" қарапайым үлгісін іске асыруға болады.
Бұл бағдарламада A-дан Z-ге дейінгі символдарды қалыптастыратын жеткізуші, сондай-ақ осы символдарды алатын тұтынушы құрылады. Бұл ретте 4 элементпен шектелген BlockingCollection<T> түрінің жиынтығы жасалады.

BlockingCollection<T> түріндегі коллекциямен жұмыс істеу үшін CompleteAdding() әдісі де пайдалы болуы мүмкін. Төменде оның хабарландыру нысаны келтірілген.

public void CompleteAdding()

Бұл әдісті шақыру жиынға ешқандай элемент қосылмайды дегенді білдіреді. Бұл IsAddingComplete сипатының true логикалық мәнін қабылдауына әкеледі. Егер жиын бос болса, онда IsCompleted сипаты true логикалық мәнін қабылдайды және бұл жағдайда Take() әдісінің қоңыраулары бұғатталмайды.

Төменде IsAddingComplete және IsCompleted сипаттарын жариялау пішіндері келтірілген.

public bool IsCompleted { get; }
public bool IsAddingComplete { get; }

BlockingCollection<T> түрінің жиыны жаңа қалыптаса бастағанда, бұл сипаттар false логикалық мәнін қамтиды. Ал CompleteAdding() әдісін шақырғаннан кейін олар true логикалық мәнін қабылдайды.

Concurrent queue

```
ConcurrentQueue<string> queue = new ConcurrentQueue<string>();
queue.Enqueue("Rob");
queue.Enqueue("Miles");
string str;
if (queue.TryPeek(out str))
Console.WriteLine("Peek: {0}", str);
if (queue.TryDequeue(out str))
Console.WriteLine("Dequeue: {0}", str);
```

Concurrent Stack

```
ConcurrentStack<string> stack = new ConcurrentStack<string>();
stack.Push("Rob");
stack.Push("Miles");
string str;
if (stack.TryPeek(out str))
Console.WriteLine("Peek: {0}", str);
if (stack.TryPop(out str))
Console.WriteLine("Pop: {0}", str);
```