Дәріс 11. Параллелизм. Parallel класы. Invoke() әдісінің көмегімен тапсырмаларды параллельдеу. For() әдісін пайдалану.

Дәрістің мақсаты: Студенттерде параллель орындалатын программалар құруда Parallel класының мүмкіндіктері туралы түсінік қалыптастыру.

Дәрісті меңгеру нәтижесінде студенттер келесі қабілеттерге ие болады:

- For(), ForEach(), Invoke() әдістерінің мүмкіндіктерін түсіну;
- For(), ForEach(), Invoke() әдістерін пайдалану синтаксисін түсіну.

Parallel класы

Parallel класы кодты қатар орындауды жеңілдетеді және параллелизмнің екі түрін: деректер мен міндеттерді ұтымды ететін әдістерді ұсынады. Parallel класы статикалық болып табылады және онда For(), ForEach() және Invoke() әдістері анықталған. Бұл әдістердің әрқайсысында түрлі нысандар бар. Атап айтқанда, For() әдісі фор таралатын циклды, ал ForEach() әдісі - foreach таралатын циклды орындайды және екі әдіс де деректердің параллелизмін қолдайды.

Ал Invoke() әдісі екі немесе одан көп әдістерді қатар орындауды қолдайды. Бұдан әрі белгілі болғандай, бұл әдістер міндеттерді немесе ағындарды нақты түрде басқаруға жүгінбей, қатар бағдарламалаудың кең таралған әдістемелерін практикада іске асыруға басымдық береді.

Invoke() әдісінің көмегімен тапсырмаларды параллельдеу

Parallel класында анықталған Invoke() әдісі оның дәлелдері түрінде көрсетілген бір немесе бірнеше әдістерді орындауға мүмкіндік береді. Ол сондай-ақ қолжетімді процессорларды пайдалана отырып, егер осындай мүмкіндік болса, кодты орындауды масштабтайды. Төменде оны хабарландырудың қарапайым нысаны келтірілген.

public static void Invoke(params Action[] actions)

Орындалатын әдістер бұрын сипатталған Action делегатымен үйлесімді болуы тиіс. Айта кетейік, Action делегаты былайша жарияланады.

public delegate void Action()

Демек, Invoke() әдісіне дәлел ретінде берілетін әрбір әдіс параметрлерді қабылдамауы немесе мәнді қайтармауы тиіс. Осы әдістің actions параметрі params түріне жататындықтан, орындалатын әдістер дәлелдердің айнымалы тізімі түрінде көрсетілуі мүмкін. Бұл мақсат үшін Action түріндегі нысандар жиымын да пайдалануға болады, бірақ көбінесе дәлелдер тізімін көрсету оңай болады.
Invoke() әдісі алдымен орындауды бастайды, содан кейін оған берілетін барлық әдістердің аяқталуын күтеді. Бұл, атап айтқанда, Wait() әдісін шақыру қажеттілігінен арылтады (және мүмкіндік бермейді). Қатар орындаудың барлық функцияларын Wait() әдісі өзіне алады. Бұл әдістердің шын мәнінде қатар орындалатынына кепілдік бермесе де, егер жүйе бірнеше процессорларды қолдайтын болса, олардың дәл осындай орындалуы болжанады. Бұдан басқа, әдістерді орындау тәртібін біріншіден соңғысына дейін көрсету мүмкіндігі жоқ және бұл тәртіп дәлелдер тізіміндегі сияқты бола алмайды.

Төменде келтірілген бағдарламаның мысалында тәжірибеде Invoke () әдісін қолдану көрсетіледі. Бұл бағдарламада () және () екі әдісі Invoke () әдісін шақыру арқылы қатар орындалады. Осы процесті ұйымдастыру қарапайымдылығына назар аударыңыз.

```
using System.Threading;
using System.Threading.Tasks;
class DemoParallel {
// Тапсырма ретінде орындалатын әдіс.
static void MyMeth() {
Console.WriteLine("MyMeth іске қосылды");
for(int count = 0; count < 5; count++) {
Thread.Sleep(500);
Console.WriteLine("MyMeth әдісіндегі санауыш мәні: " + count );
}
Console.WriteLine("MyMeth аяқталды");
}
// Тапсырма ретінде орындалатын әдіс.
static void MyMeth2() {
Console.WriteLine("MyMeth2 іске қосылды");
for(int count = 0; count < 5; count++) {
Thread.Sleep(500);
Console.WriteLine("MyMeth2 әдісіндегі санауыш мәні: " + count );
}
Console.WriteLine("MyMeth2 аяқталды");
}
static void Main() {
Console.WriteLine("Негізгі ағын іске қосылды.");
// Екі атаулы әдісті параллель орындау.
Parallel.Invoke(MyMeth, MyMeth2);
Console.WriteLine("Негізгі ағын аяқталды.");
}
}
```

For() әдісін пайдалану

Деректердің параллелизмі, атап айтқанда, Parallel класында анықталған For() әдісінің көмегімен қолдау табады. Бұл әдіс бірнеше пішіндерде бар. Оны қарауды төменде келтірілген ең қарапайым формадан бастаймыз:

public static ParallelLoopResult For (int fromInclusive, int toExclusive, Action<int> body)

мұнда fromInclusive циклді басқару айнымалысына сәйкес келетін нәрсенің бастапқы мәнін білдіреді; ол сондай-ақ итерациялық, немесе индекстік, мән деп аталады; a toExclusive - түпкілікті мәннен бір бірлікке артық мән. Циклдің әрбір қадамында циклді басқару айнымалы бірлікке ұлғаяды. Демек, цикл бастапқы мәннен минус бірлікке қарай біртіндеп жылжиды. Циклдік орындалатын код body параметрі арқылы берілетін әдіспен көрсетіледі. Бұл әдіс келесі түрде жарияланатын Action<int> делегатымен үйлесімді болуы тиіс.

public delegate void Action<in T>(T obj)

For() әдісі үшін T жиынтықталған параметрі, әрине, int түрі болуы тиіс. obj параметрі арқылы берілетін мән циклді басқару айнымалысының келесі мәні болады. Ал body

параметрі арқылы берілетін әдіс атаулы немесе жасырын болуы мүмкін. For() әдісі циклдің аяқталу күйін сипаттайтын ParallelLoopResult түріндегі нысанның данасын қайтарады. Қарапайым циклдер үшін бұл мәнді елемеуге болады.

For() әдісінің басты ерекшелігі - ол мұндай мүмкіндік болған кезде циклдегі кодтың орындалуын параллельдеуге мүмкіндік береді. Ал бұл өз кезегінде өнімділіктің артуына алып келуі мүмкін. Мысалы, циклдегі жиымды түрлендіру процесі жиымның әртүрлі бөліктері бір мезгілде түрленетіндей етіп бөліктерге бөлінуі мүмкін. Алайда, өнімділіктің артуына әртүрлі орындау ортасындағы қолжетімді процессорлардың санындағы айырмашылықтарға байланысты, сондай-ақ ұсақ циклдерді бөлшектеу үнемделген уақыттан асатын шығындарды құрауы мүмкін болғандықтан кепілдік берілмейтінін ескеру қажет.

Төменде келтірілген бағдарламаның мысалында іс жүзінде For() әдісін қолдану көрсетіледі. Осы бағдарламаның басында 1000000000 бүтін мәннен тұратын data жиымы жасалады. Содан кейін For() әдісі шақырылады, оған циклдің "денесі" ретінде MyTransform() әдісі беріледі. Бұл әдіс data жиымында еркін түрлендірулерді орындайтын бірқатар операторлардан тұрады. Оның мақсаты - нақты операцияны бейнелеу. Төменде біршама егжей-тегжейлі түсіндірілгендей, деректердің параллелизмі қандай да бір оң нәтиже беруі үшін орындалатын операция әдеттен тыс болуы тиіс. Олай болмаған жағдайда циклды жүйелі орындау тезірек аяқталуы мүмкін.

```
using System;
using System.Threading.Tasks;
class DemoParallelFor {
static int[] data;
// Параллель орындалатын цикл тұлғасы ретінде қызмет ететін әдіс.
static void MyTrknsform(int i) {
data[i] = data[i] / 10;
if(data[i] < 10000) data[i] = 0;
if(data[i] > 10000 & data[i] < 20000) data[i] = 100;
if(data[i] > 20000 & data[i] < 30000) data[i] = 200;
if(data[i] > 30000) data[i] = 300;
}
static void Main() {
Console.WriteLine("Негізгі ағын іске қосылды.");
data = new int[100000000];
// Мәліметтерді әдеттегі for циклында инициалдау.
for(int i=0; i < data.Length; i++) data[i] = i;
// Циклді For() әдісімен параллель күйге келтіру.
Parallel.For(0, data.Length, MyTransform);
Console.WriteLine("Негізгі ағын аяқталды.");
}
}
```

ForEach() әдісін пайдалану

```
using System;
using System.Threading.Tasks;
class DemoParallelForWithLoopResult {
static int[] data;
// Параллель орындалатын цикл тұлғасы ретінде қызмет ететін әдіс.
static void DisplayData(int v, ParallelLoopState pls) {
```

```
// Теріс мән табылған жағдайда цикл жұмысын тоқтату.
if(v < 0) pls.Break();
Console.WriteLine("Мән: " + v);
}
static void Main() {
Console.WriteLine("Негізгі ағын іске қосылды.");
data = new int[100000000];
// Мәліметтерді инициалдау.
for(int i=0; i < data.Length; i++) data[i] = i;
// Теріс мәнді data жиымына ауыстыру,
data[100000] = -10;
// ForEach() әдісінің көмегімен параллель орындалатын циклді пайдалану
ParallelLoopResult loopResult = Parallel.ForEach(data, DisplayData);
// Циклдің аяқталғанын тексеру.
if(!loopResult.IsCompleted)
Console.WriteLine("\nЦикл мерзімінен бұрын аяқталды, себебі циклдің " +
loopResult.LowestBreakIteration + "нөмірлі қадамында теріс мән табылды\n");
Console.WriteLine("Негізгі ағын аяқталды.");
}
}
```