Дәріс 3. Ағынға мәліметтерді беру. Ағындардың басымдықтары (приоритет). Ағындарды синхронизациялау.

Дәрістің мақсаты: Студенттерде ағымдардың өзара байланысты жұмысын басқарумен байланысты дағдыларды қалыптастыру.

Дәрісті меңгеру нәтижесінде студенттер келесі қабілеттерге ие болады:

- Ағынға мәліметтерді беру тәсілін түсіну;
- Ағынның басымдылығын орнату тәсілдерін түсіну;
- Ағындарды синхронизациялау ұғымын түсіну.

Ағынға аргумент беру

Бастапқыда .NET Framework ортасында дәлелді ағынға беруге болмайды, ол басталғанда, себебі ағынға кіру нүктесі ретінде қызмет ететін әдіс параметрлері болуы мүмкін. Егер ағынға қандай да бір ақпарат беру талап етілсе, бұл мақсатқа әртүрлі айналма жолдармен баруға тура келді, мысалы жалпы айнымалысы. Бірақ бұл кемшілік кейіннен жойылды, енді аргумент ағынға жіберілуі мүмкін. Бұл үшін басқаларды пайдалануға тура келеді Start() әдісінің, Thread класының құрастырушысының, сондай-ақ ағынға кіру нүктесі ретінде.
Аргумент Start() әдісінің келесі пішінінде ағынға жіберіледі.

public void Start (object параметрі)

Параметр дәлел ретінде көрсетілетін нысан автоматты түрде жіберіледі ағынға кіру нүктесінің рөлін атқаратын әдіс бойынша. Демек, беру үшін ағынға дәлел, оны Start () әдісіне беру жеткілікті.
Start () әдісінің параметрленген нысанын қолдану үшін мыналар қажет Thread сыныбының құрастырушы пішіні:

public Thread(ParameterizedThreadStart запуск)

мұнда іске қосу ағынды орындауды бастау мақсатында шақырылатын әдісті білдіреді.

IsBackground қасиеті

Жоғарыда айтылғандай, .NET Framework ортасында басым және фондық ағындардың екі түрі анықталған. Олардың арасындағы бірден-бір айырмашылық - үдеріс басым ағын аяқталғанға дейін аяқталмайды, ал фондық ағындар барлық басым ағындар аяқталғаннан кейін автоматты түрде аяқталады. Әдепкі құрылатын ағын басымдыққа ие болады. Бірақ оны сыныпта анықталған IsBackground сипатын пайдалана отырып, фондық етуге болады Thread, келесі түрде.

public bool IsBackground { get; set; }

Ағынды өңдік ету үшін IsBackground сипатына логикалық мән беру жеткілікті. Ал false логикалық мәні ағынның басым екенін көрсетеді.

Ағындар басымдықтары

Ағын басымдығын орнату үшін келесі қасиет қолданылады:

public ThreadPriority Priority{ get; set; }

Қасиеттің типі тізбе болып табылады, тізбеде басымдықтың келесі мәндері анықталған:

ThreadPriority.Highest
ThreadPriority.AboveNormal
ThreadPriority.Normal
ThreadPriority.BelowNormal
ThreadPriority.Lowest

Мысал 1. Ағындардың басымдықтарын басқару мысалы.

```
using System;
using System.Threading;
class MyThread {
public int Count;
public Thread Thrd; // ағындық типтегі объект
static bool stop = false;
static string currentName;

public MyThread(string name) {
Count = 0;
Thrd = new Thread(this.Run); // ағынды құру
Thrd.Name = name;
currentName = name;
}
// Ағынға кіру нүктесі.
void Run() {
Console.WriteLine(Thrd.Name + " ағыны іске қосылды.");
do {
Count++;
if(currentName != Thrd.Name) {
currentName = Thrd.Name;
Console.WriteLine(currentName + " ағымы орындалуда");
}
} while(stop == false && Count < 1000000000);
stop = true;
Console.WriteLine(Thrd.Name + " ағымының жұмысы аяқталды.");
}
}
class PriorityDemo {
static void Main() {
MyThread mt1 = new MyThread("жоғары басымдықты");
MyThread mt2 = new MyThread("төмен басымдықты");
// Ағындар басымдықтарын орнату
mt1.Thrd.Priority = ThreadPriority.AboveNormal;
mt2.Thrd.Priority = ThreadPriority.BelowNormal;
// Ағындарды іске қосу
mt1.Thrd.Start();
mt2.Thrd.Start();
mt1.Thrd.Join();
mt2.Thrd.Join();
Console.WriteLine();
Console.WriteLine(mt1.Thrd.Name + " ағыны " + mt1.Count + " дейін санады");
```

```
Console.WriteLine(mt2.Thrd.Name + " ағыны " + mt2.Count + " дейін санады");
}
}
```

Синхронизация

Бірнеше ағындар пайдаланылғанда, кейде екі немесе одан да көп ағындардың әрекеттерін үйлестіруге тура келеді. Мұндай үйлестіруге қол жеткізу процесі үндестіру деп аталады. Синхрондауды қолданудың ең көп тараған себебі екі немесе одан да көп ағындардың арасында бір мезгілде бір ғана ағынға қолжетімді болатын жалпы ресурсты бөлу қажеттілігі. Мысалы, бір ағымда ақпаратты файлға жазу орындалғанда, екінші ағынға оны дәл сол уақытта жасауға тыйым салынуы тиіс. Егер бір ағын басқа ағынмен шақырылатын оқиғаны күтсе, үндестіру қажет. Мұндай жағдайда басқа ағымда оқиға болғанға дейін ағындардың бірін тоқтата тұруға мүмкіндік беретін қандай да бір қаражат талап етіледі. Бұдан кейін күтуші ағын өз орындауын жалғастыра алады.

Синхрондау негізіне блоктау ұғымы қойылған, ол арқылы объектідегі кодтық блокқа қол жеткізуді басқару ұйымдастырылады. Нысан бір ағынмен құрсауланғанда, қалған ағындар құрсауланған кодтық блокқа қатынаса алмайды. Бұғаттау бір ағынмен алынғанда, нысан басқа ағымда пайдалану үшін қол жетімді болады.

Құрсаулау құралы C# тіліне кірістірілген. Осының арқасында барлық нысандар үндестірілуі мүмкін. Үндестіру lock кілт сөзінің көмегімен ұйымдастырылады.

Ол C# бағдарламасында басынан бастап қарастырылған болатын, сондықтан оны пайдалану бір қарағанда әлдеқайда оңай. Шын мәнінде объектілерді синхрондау көптеген бағдарламаларда C# -те іс жүзінде байқалмай жүреді.

Төменде бұғаттаудың жалпы нысаны келтірілген:

```
lock(lockObj) {
//үндестірілетін операторлар
}
```

мұнда lockObj үндестірілетін нысанға сілтемені белгілейді. Егер бір ғана оператор үндестірілсе, онда мәнерлі жақшаның қажеті жоқ. lock операторы осы нысан үшін бұғаттаумен қорғалған код элементінің тек осы құрсаулауды алатын ағында пайдаланылатынына кепілдік береді. Ал қалған барлық ағындар бұғаттау алынып тасталғанға дейін бұғатталады. Бұғаттау ол қорғайтын код үзіндісі аяқталғаннан кейін алынады.