**Threadler**

Umumiyyetle iki proses zamani user app den narahat ola biler

* *Intensive hesablama*
* *Input/output goturulme emeliyyati*

Bu iki proses zamani patok agir ise dusduyu zaman UI block ola bilir ve bu da app in responsivliyini itirir.

Bunu aradan qaldirmaq ucun Yeni thread (patok) ehtiyac duyulurki.Main thread donmadan basqa bir thread prosesi gorsun ve daha sonra app responsive sekilde isini davam etdirsin.

Bunlar mueyyen derecede ise yarasada artiq bize 2 -3 den cox patok lazim oldugu zaman biz hemin sayda thread yaratmali oluruq.Onsuzda threadin yaranmagi ozu ayri problemdir.Cunki bir threadin yaranmagi cox vaxt ve resources teleb eliyir.Ona gore threadlerden istifade artiq duzdgun sayilmir.Bunun Microsoft on gorur ve **threadpool** deye bir mexanizm yaradir.

**Thread pool isleme ardiciligi**

QueueUserWorkItem metodunu cagiraraq thread pool da novbeye bir thread elave edirik.Eger threadpoolda available thread varsa o cagirilir amma eger yoxdursa ve threadpoolda max thread sayi qeder thread yoxdursa bu zaman yeni thread yaranir ve novbeye dusur.

QueueUserWorkItem metodunu cagirdigimiz zaman hemin prosesler olur ve bir thread ise dusur,problem burdadirki Threading isi ne zaman bitecek bilinmir .Join metodu yoxdur.Bu problem hell etmek ucun Signal mexanizmden istifade etmek lazimdi.

ThreadPool classini tez tez istifade etmeyeceyik

**Differences create manually thread and take thread from threadpool**

Thread pool daki butun threadler background threaddir.Baxmayaraq bizim el ile yaratdigimiz threadler foreaground thread olurlar.(biz background ede bilerik).

Threadpooldaki threadlere mudaxile ede vey a abort ede bilmerik.Ancaq oz yaratdiqlarimiza ede bilerik.

Threadpool daki threadlere join ede bilmerik.

Thread pooldaki threadler yeniden istifade ede bilinirler.Lakin manual yaratdigimiz threadler istifadeden sonar destroy olurlar.

Basqa ferglerde var ancaq esas bilmeli oldugumuz bunlardir.

**Unblocking UI(USER INTERFACE)**

Agir proseslerinin toretdiyi en boyuk problem UI donmasidir.Buna sebeb bu agir prosesler uzunden UI update etmeye gucu catmir threadin bu sebebden bu agir prosesleri basqa threaded etmek lazimdirki UI da donma hali olmasin.

.net 4.5 app olanlar zaman async/await den istifade bu problem hell edecekdir. Asagi prioritet deki net app ler ucun ise proqramci ozu manual olaraq mudaxile etmelidir.

**Working with the Task Parallel Library**

Threadin en boyuk problem intensive resurs yemeyidir.Bir thread ise dusen zaman o digger thread le run olmaq ucun CPU ugrunda savasir.Bu en boyuk problemdir.

.NET 4 ile Microsoft *Task* classini tqedim etdi,hansi ki asinxron operation numayis etdirir.Task ozu threadpooldaki threadleri istifade edir,LAKIN flexibility ve nece task yaranip kimi bizim threaddeki problemlerin eksini teklif edir.

Iki cur Task classi vardir

Task

Task<TResult>

Eger bir netice qaytaramaya ehtiyac yoxdursa 1 ci eksinedirse 2 cini istifade etmleiyik.

Task yaratmagin bir nece yolu vardir.

* Task classinin instanceni yaradaraq.
* Taskfactory.StartNew
* Task.Run
* Ve ya countination Task ile

**TASK SCHEDULER**

Tasklarin threaded isleme novbesine **task scheduler** nezaret edir.Normalda schedulerden istifade etmirik nese specific bir shey gostermesek oz default deyerini verecekdir.

**Using The Parallel Class**

Bildiyimiz kimi Task lar thread lerin daha yaxsi versiyasidir ve tasklar ozu arxada threadpooldan threadler isledir.Baxmayaraqki tasklar threadlerden daha yunguldur bezen daha da yaxsi bir process lazim olur multitask ucun.Microsoft bunun ucun Parallel clasini yaratdi.Bu classin ozu mumkun qeder parallellik yaradir lakin 100% parallel qarantisini vermir.Paralele yaxin.

**WORKING WITH CONTUNATIONS**

Tamam tasklarla vey a parallellerle islemeyi bacardiq ,bes her hansi bir metod bitmemis digerinin islememesi ucun ne etmeliyik.Burada bir nece yolumuz var.

Meselen ferz edekki 3 metodumuz var Step1() Step2() Step3()

**ContinueWith()** metodundan istifade ede bilerik.

Step1 ve step2 bitmemis step3 baslamasin.Bu zaman task.factoryden istifade edip

**ContinueWhenAll()** metodundan istifade ede bilerik.

Step1 ve ya step 2 bitmemis step3 baslamasin.Bu zaman task.factory classindan istifade edip

**ContinueWhenAny()** metodundan istifade ede bilerik