موضوع الدرس : _____ التاريخ : / /

Asynchorous Programming

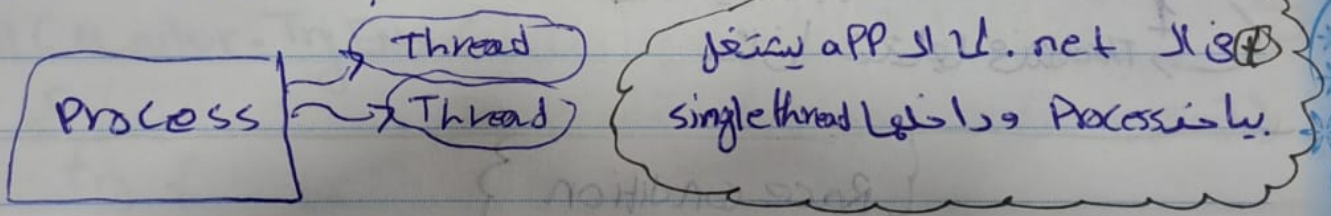
Video 01 شعير عبد النبي

* need to deal with running different units out-of-order

* any application starts on windows ⟹ Process with id is created

(ByDesign) CLR create a single thread to excute code

Thread ⟹ Basic unit in which the operation system allocates processor time.



---

* by default all threads are Forground not Background

* ال Forground هوا اللي بيخلي app عايش
ال Background بيموتوا for بيموتوا من OS

---

t1.Start(); ⟶ اشغله

t1.Name(); ⟶ اطبع اسمه

t1.ThreadState(); ⟶ Running , Unstarted الحالة

التاريخ : / /

موضوع الدرس :

ازاى أعرّف thread

```
Thread T1 = new Thread( );                    (1)
Thread T2 = new Thread(new ThreadStart());    (2)
```

---

* لما بعرف 2 Thread بيشتغلوا Parallel
  واحدهم start ↓

T1.Join(); ⟸ منستنا T1 لحد ما يخلص الأول
T2.Start(); ⟸ ابدأ

بلمتى استخدم الJoin

لو عايز النتيجة اللى جاية من Thread 1 عشان استخدمها فى Thread2

Race condition

لو الـ method فيها Parameters

```
Thread T1 = new Thread( () => wallet.Debit(50) );
```

Lambda ⟵ Thread لأ

---

is a scenario where that come of the Program is affected because of timing

الحل ⟸ LOCK

التاريخ : / /

موضوع الدرس :

lock ( ) → ref أي type

↳ Private readonly object bitcoinLock = new object();

{
}

اذا دخل الـ T1 اذا الـ lock بيقفل لحدما يخرج وبعدين بيفتح تاني و تدخل الـ T2 ومتطبقش عشان الشرط

Deadlock

when two or more ~~methods~~ threads are frozen in their excution because they are waiting for each other to finish.

manager class

used to manage set of objects of another class.

(*) استخدم الـ (1) monitor class

(*) if (Monitor.TryEnter(_To, 1000))

Time out ← ~~if (Monitor.TryEnter(_To))~~

```
{
  try {
    العمليات المحصلها
  }
  catch {
  }
  finally {
    monitor.Exit(to);
```

(*) else

lock ومعملش

(*) (2) الترتيب المسبق

بيدل مايبس الـ to from على الـ locks

راجع الكود

التاريخ : / /

موضوع الدرس :

Thread Pool

creating thread has overhead in time and memory → Pre-created recyclable threads

* always background

الطريقة الأولى

ThreadPool.QueueUserWorkItem(new WaitCallBack( ));

* Void method with Parameter type => object

الطريقة الثانية

Task.Run( )

method

video02

Video02

~~Task.Run()~~

* القيم الافتراضية لأي thread

* isPooled = false

* IsBackground = False

* لو بدي Thread !! Paramaterized method

new Thread(() => Display("a")

lambda exp

كيف بدي اطبق الـ Task Class

الصفحة الى بعدها

الأمل

Task.Run(() => Display(" mytask")

الجملة دى شبه جملة var Th = new Th الى ؟ ص ٥ ولكن في فروقات

(*) IsPooled = True

(*) IsBackground = True

(*) لما كنت بعوز استنى ال Thread لحد ما يخلص كنت بستخدم .join()

(*) في ال task ⟵ .Wait()

(*) ال task كلها Thread Pool إذا دائما Background

كيف ممكن ال task يرجع قيمة ؟

task<> | Task< DateTime > task = Task.Run(() => ...);
task | CW (Task); → w as object

(*) objects اسمه Result جوا ال Task
بـ Run ال function ويرجع الناتج

(*) CW (task.Result) ① (*) يتقل ال Block ال thread
until result is ready

(*) CW(task.GetAwaiter().GetResult()); ②
نفس الناتج

(*) إذا ال task ممكن يرجع قيمة ودى
نقطة تفوق على ال thread

long running tasks

don't use ThreadPool (long) وقت البدء > وقت التنفيذ
use ThreadPool (short) وقت البدء < وقت التنفيذ

كيف ممكن اعمل Longrunning task

* مش عاوزه task في ThreadPool

(*) var task = Task.Factory.StartNew(() => ~~~);
(*) TaskCreationOption.LongRunning

(*) لو عملت ▶ هلاقي IsPooled = False

Exception Propagation

(*) لو عملت try و catch و حطيت جوا الـ try ⟸
var th = new Thread(ThrowException);
وجوا الـ catch
⟶ cw("ex is thrown");

(*) مش هيعمله عشان الـ thread المسؤول عن try و catch
مختلف عن الـ thread اللي بينفذ الـ method

(*) الحل اني احط جزء الـ try والـ catch
جوا الـ method نفسها

(*) في بعض الحالات بحتاج اعمله Exception Propagat

(٧) بستخدم ال task

(٨) في try و catch في ال main

جوا ال try

```
Task.Run( ).wait();
```

Task Continuation

مشكلة task.result

انها متعمل block لل thread لحد ما ال result تطلع

بنحل إزاي

(١) ① انا بعرف

```
Task task = Task.Run(
  () => GetPrimes(2, 3_000_000));
```

② ② بعرف

```
var awaiter = task.GetAwaiter();
```

TaskAwaiter <TResult>

③ بعد كده

```
awaiter.OnComplete( () => awaiter.GetResult()
  cw( " ");
```

دى بتعمل block برضو بس مش هينفذها إلا بعد onComplete

on complete();

في طريقة آخرى

```
task.ContinueWith( (x) => cw(x.result) );
```

التاريخ : / /

موضوع الدرس :

Task.Delay() Vs Thread.Sleep()

* لو عملت Call لـ DelayUsingTask(5000); ✱
* هيرن على طول اللي بعده ✱
* SleepUsingThread(5000) ✱
* هينتظر 5 ثواني ✱

ليه؟ ⇐ علشان الـ run بتعمل Task

```
Task.Delay(ms).GetAwaiter().OnCompleted(() =>
    CW("كوسة"));
```

Synchronous & Asynchronous

✱ Sync ⇒ does its work before return to the caller
Task1 ✱ ينتظر Task2 حتى ينتهي

✱ Asyn ⇒ can do most or all of its work after returning to the caller ✱

المهم في الكود راجع الكود

Async Function

(*) The await keyword simplifies attaching of continuations

is Promising to return ← value

Guidelines

① نهاية اسم الـ method Async

② الـ method نوعها async

③ non-blocking code أي لازم اسبقه بكلمة await

```
static async Task<string>
ReadContent(url) {
var client = new HTTPClient();
await client.GetStringAsync(url);
```

~~لما اجي اناديه~~

(*) طالما موجودة كلمة async اذا كلمة await كلمة محجوزة

لما اجي اناديه

```
var content = await ReadContentAsync("url");
```

(*) الـ main هتبقى ⇐ static async Task Main

التاريخ : / /

موضوع الدرس :

CancellationToken

* القدرة على كنسلة التاسك في اى وقته

Task — thread — وزع تعمل كود معين

CancellationTokenSource

* يوفر ال cancellation Infrastructure

أفضل استخدام ليها مع ASP وع Framework withAll

* cts.Cancel();

* while(!cts.Token.IsCancellationRequested) (1)

هل حصل cancel ولا؟

بدلها

* while(true) (2)
await Task.Delay(4000, cts.Token)

* 1 استفدت ايه؟

في الأول هيستنى 4 ثواني لغاية ما كمل التاسك او cancel وصلت فوق

بدل ده

* بحط ال catching try في while

```
while(){
cts.Token.ThrowIfCancellationRequested();
}
```

الأمر

11 Report a Progress from Async

راجع الكود الموضوع كله عبارة عن كود

Combinators

* بتقلل ال Performance
* لو عندك تاسكين ويكفي واحد منهم بخلص ⇐ بستعمل whenany

* في اليونيتوب لازم شرطين لازم يتحققوا حتى تقدر اليونيتوب يعمل

① Task.FromResult("a7a") بدل من Task.Run(() => "a7a");

② Task.CompletedTask ⟹ void Task

WhenAny() أي واحد من التاسكات دول يخلص الأول يطبع

WhenAll(); ⟹ Array of Tasks بترجع

Asyncronous Patterns

* any Async return Task Task<T>
* Has "Async" Suffix except / Task combinators
* Cancellation AND/OR progress reporting
* Return quickly to caller synchronous Phase
* Does not tie up a Thread if I/O Bound.

الأمل

concurency & Parallelism

التعامل مع الكثير من الأشياء في وقت واحد

Task1 → Task2

time

Single core

* عادة التنفيذ على

core #1

التعامل المتزامن للكثير من الأشياء وقته واحد

— task1 —

— task2 —

time

core #1 | core #2

Parallel.ForEach( );

await & async

درس مختلف عن

عبارة عن كود تجربته وكتبته خلاص

multiThreading Basics | Passionate Coders

main thread = aT process

* الـ thread إلي بيقوم في الاول خالص في أي application

* لو عندك كود بياخد وقت طويل في التنفيذ وانت مش عايز الـ

بيستناه بتستعمل الـ multithreading ,

الـ method إلي بتتنفذ داخل الـ thread يا

(1) يا تاخذ ولا Param (2) بتاخذ Param واحد بس نوعه object

التاريخ : / / موضوع الدرس :

(*) وأنت بتشتغل multithreading حاول متعتمدش على Static resources

في المثال بتعنا عشان الأرقام ميطلعش فيها أي بيض

(*) أعرف new object واعمل lock واحط جواه /

الكود بتاعي

```
Private static object _a = new();
lock (_a) {
    my code;
}
```

(*) مفهوم دا وقت في 2 threads كل واحد جواه كود لما أحط كل كود من دول جوا lock الـ thread التاني مش هينفذ الكود الي جوا الـ lock إلا لما يتأكد ان الـ Lock مفتوح

Thread Priority

thl. Priority = ThreadPriority. Highest ; / Lowest ();

(*) الـ multithreading هدفها تحسين استجابة الـ application

Background threads لما كل Forground thread تخلص الـ app بيقفل حتى لو الـ Background لسه شغال

لو هتتعامل مع الـ Thread اتعامل مع الـ Thread Pool أي Thread Pool يبقى Background

ThreadPool. QueueUserWorkItem (ProcessBatch, object

object? state ⇐ ممكن استعملها في الـ Thread cancellation

يدير عملية انشاء الـ thread والغاؤه استخدامها

Thread Cancellation

```
Var cts = CancellationTokenSource();
cts.Cancel
```

يستخدمها في مثلا

```
ThreadPool.~~~~(ProcessBatch(), cts.Token)
```

```
if (CancelToken.IsCancellationRequested)
    cts.Cancel();
```

فلما أعمل كدة

Thread يقف

Video 02

T A P

Task-Based Asynchronous Pattern

* Synchronous
  Blocking

* Asynchronous
  no Blocking

Task.Run(▭)

↳ جوا دا هنا

بتعتمد على الـ thread Pool

وبحطها في thread تاني لوحده

* لو حابب تستخدم الكلاس

الطريقة الأكثر استعمالاً

1. اللي الـ method

Task ⇐ return

* Task.CompletedTask; ⇒ بترجع تاسك حالتها مكتمل Dummy

الأمثل

② Private static async Task_ _ _ _

طالما كتبت كده - يبقى مش return حاجة

③ ✗ عشان استنى تاسك لحد لما تخلص await Task.Delay();

وعشان اكتب await لازم تكون جوا async method

أول ما يوصل لكلمة await بعمل thread جديد وبعمله block

ويرجع التحكم لل caller (main thread)

لو عايز await كذا Task

await Task.WhenAll(task1, task2);

await Task.WhenAny();

لو عايز { لو عايز أشغل أول تاسك ولما تخلص أشغل واحد التانية

var task2 = task1.ContinueWith(...);

لو عندي Task<int>
وعايز آخد القيمة اللي
لازم await

async t => await ProcessBatch2(cts.token)

var x = await Batch...
↓
int

لو عايز آخد result نفسها بعمل await

var x = await ___
↓
int

Continuation

var x = Batch ___
↓
Task<int>

التاريخ : / / 16

موضوع الدرس :

* أقدر أوصل ليها Status

task.Status

* لو مستخدم await وأدي method بعد الـ cancel مش هيحصل الـ cancellation token.

* لو مستعمل await على method هيرمي فيها الـ cancel Ex لازم أحط جزء الـ cancellation في try و catch

```
try {
    await task1;
}

catch (OperationCanceledException e) { }
```