عنوان: **اصول برنامه نویسی موازی درNET. نسخه 4 بخش اول - 1**
نویسنده: اشکان 56
تاریخ: ۱۳۹۱/۰۳/۳۱ ۱۲:۷
آدرس: www.dotnettips.info
برچسب‌ها: C#, Task Programming Language, Threading, Parallel Extensions

بدون هیچ مطلب اضافی به سراغ اولین مثال می‌رویم. قطعه کد زیر را در نظر بگیرید :

```
using System;
using System.Threading.Tasks;

namespace Listing_01 {

class Listing_01 {

static void Main(string[] args) {
  Task.Factory.StartNew(() => {
         Console.WriteLine("Hello World");
  });

  // wait for input before exiting
  Console.WriteLine("Main method complete. Press enter to finish.");
  Console.ReadLine();
 }
}
```

در کد بالا کلاس Task نقش اصلی را بازی می‌کند.این کلاس قلب کتابخانه برنامه نویسی Task یا Task Programming Library می‌باشد.

در این بخش با موارد زیر در مورد Taskها آشنا می‌شویم:

- ایجاد و به کار انداختن انواع مختلف Task ها.
- کنسل کردن Task ها.
- منتظر شدن برای پایان یک Task.
- دریافت خروجی یا نتیجه از یک Task پایان یافته.
- مدیریت خطا در طول انجام یک Task

خب بهتر است به شرح کد بالا بپردازیم:

رای استفاده از کلاس Task باید فضای نام System.Threading.Tasks را بصورت ریر مورد استفاده قرار دهیم.

```
using System.Threading.Tasks;
```

این فضای نام نقش بسیار مهمی در برنامه نویسی Taskها دارد . فضای نام بعدی معروف است : System.Threading . اگر با برنامه نویسی تریدها بروش مرسوم وکلاسیک آشنایی دارید قطعاً با این فضای نام آشنایی دارید. اگر بخواهیم با چندین Task بطور همزمان کار کنیم به این فضای نام نیاز مبرم داریم. پس :

```
using System.Threading;
```

خب رسیدیم به بخش مهم برنامه :

```
Task.Factory.StartNew(() => {
   Console.WriteLine("Hello World");
});
```

متد استاتیک Task.Factory.StartNew یک Task جدید را ایجاد و شروع می‌کند که متن Hello Word را در خروجی کنسول نمایش می‌دهد. این روش ساده‌ترین راه برای ایجاد و شروع یک Task است.

در بخش‌های بعدی چگونگی ایجاد Taskهای پیچیده‌تر را بررسی خواهیم کرد . خروجی برنامه بالا بصورت زیر خواهد بود:

```
Main method complete. Press enter to finish.

Hello World
```

روشهای مختلف ایجاد یک Task ساده :

- ایجاد کلاس Task با استفاده از یک متد دارای نام که در داخل یک کلاس Action صدا زده می‌شود. مثال :

```
Task task1 = new Task(new Action(printMessage));
```

استفاده از یک delegate ناشناس (بدون نام). مثال :

```
Task task2 = new Task(delegate {
   printMessage();
});
```

- استفاده از یک عبارت لامبدا و یک متد دارای نام . مثال :

```
Task task3 = new Task(() => printMessage());
```

- استفاده از یک عبارت لامبدا و یک متد ناشناس (بدون نام). مثال :

```
Task task4 = new Task(() => {
   printMessage();
});
```

قطعه کد زیر مثال خوبی برای چهار روشی که در بالا شرح دادیم می‌باشد:

```
using System;
using System.Threading.Tasks;

namespace Listing_02 {

class Listing_02 {

static void Main(string[] args) {

   // use an Action delegate and a named method
   Task task1 = new Task(new Action(printMessage));

   // use a anonymous delegate
   Task task2 = new Task(delegate {
   printMessage();
});

  // use a lambda expression and a named method
  Task task3 = new Task(() => printMessage());

  // use a lambda expression and an anonymous method
  Task task4 = new Task(() => {
    printMessage();
  });
  task1.Start();
  task2.Start();
  task3.Start();
  task4.Start();

  // wait for input before exiting
```

```
  Console.WriteLine("Main method complete. Press enter to finish.");
  Console.ReadLine();
 }

 static void printMessage() {
  Console.WriteLine("Hello World");
  }
 }
}
```

خروجی برنامه بالا بصورت زیر است :

```
Main method complete. Press enter to finish.

Hello World

Hello World

Hello World

Hello World
```

نکته 1 : از مند استاتیک Task.Factory.StartNew برای ایجاد Task هایی که رمان اجرای کوتاه دارند استفاده می‌شود.

نکته 2 : اگر یک Taskدر حال اجرا باشد نمی‌توان آنرا دوباره استارت نمود باید برای یک نمونه جدید از آن Task ایجاد نمود و آنرا استارت کرد.

نظرات خوانندگان

نویسنده: رحمت رضایی
تاریخ: ۱۷:۵۱ ۱۳۹۱/۰۳/۳۱

از مند استاتیک Task.Factory.StartNew برای ایجاد Task هایی که زمان اجرای طولانی هم دارند استفاده می‌شود :

```
Task.Factory.StartNew(() =>
            {
                Thread.Sleep(1000);
            }, TaskCreationOptions.LongRunning);
```

نویسنده: ali
تاریخ: ۱۸:۳۷ ۱۳۹۱/۰۳/۳۱

سلام
مرسی از آموزشتون

این روش چه برتری نسبت به شیوه کلاسیک موازی کاری داره ؟
آیا همه امکانات شیوه کلاسیک رو پوشش می‌ده ؟

بی صبرانه منتظر ادامه آموزش هستم.
پیروز باشید.

نویسنده: حسین مرادی نیا
تاریخ: ۱:۵۹ ۱۳۹۱/۰۴/۰۱

اگر منظور شما از روش‌های کلاسیک استفاده از Threadهاست باید بدانید که آن روش‌ها برای CPUهای تک هسته ای در نظر گرفته شده بودند. همانطور که می‌دانید در CPUهای تک هسته ای ، CPU تنها قادر به اجرای یک وظیفه در یک واحد زمان می‌باشد. در این CPUها برای اینکه بتوان چندین وظیفه را همراه با هم انجام داد CPU بین کارهای در حال انجام در بازه‌های زمانی مختلف سوییچ میکند و برای ما اینطور به نظر می‌آید که CPU در حال انجام چند وظیفه در یک زمان است.
اما در CPUها چند هسته ای امروزی هر هسته قادر به اجرای یک وظیفه به صورت مجزا می‌باشد و این CPUها برای انجام کارهای همزمان عملکرد بسیار بسیار بهتری نسبت به CPUهای تک هسته ای دارند.
با توجه به این موضوع برای اینکه بتوان از قابلیتهای چند هسته ای CPUهای امروزی استفاده کرد باید برنامه نویسی موازی (Parallel Programming) انجام داد و روش‌های کلاسیک مناسب این کار نمی‌باشند.

نویسنده: محمد
تاریخ: ۹:۴۸ ۱۳۹۱/۰۴/۰۱

ممنون

نویسنده: saleh
تاریخ: ۰:۱۲ ۱۳۹۱/۰۴/۱۷

شما در کد خودتون taskها را قبل از دستور چاپ متن main method ... نوشته و استارت داده بودید ولی در خروجی برعکس این موضوع اتفاق افتاده! میشه درموردش توضیح بدید؟

نویسنده: وحید نصیری
تاریخ: ۰:۳۰ ۱۳۹۱/۰۴/۱۷

توضیحات بیشتر در اینجا

عنوان: **اصول برنامه نویسی موازی درNET. نسخه 4 بخش اول - 2**
نویسنده: اشکان 56
تاریخ: ۱۳۹۱/۰۴/۰۱ ۱:۱۹
آدرس: www.dotnettips.info
برچسب‌ها: C#, Task Programming Language, Threading, Parallel Extensions

تنظیم وضعیت برای یک Task

در مثال ذکر شده در قسمت قبل هر چهار Task یک عبارت را در خروجی نمایش دادند حال می‌خواهیم هر Task پیغام متفاوتی را نمایش دهد.برای این کار از کلاس زیر استفاده می‌کنیم :

```
System.Action<object>
```

تنظیم وضعیت برای یک Task این امکان را فراهم می‌کند که بر روی اطلاعات مختلفی یک پروسه مشابه را انجام داد.

مثال :

```
namespace Listing_03 {
class Listing_03 {
  static void Main(string[] args) {
   // use an Action delegate and a named method
   Task task1 = new Task(new Action<object>(printMessage),"First task");

   // use an anonymous delegate
   Task task2 = new Task(delegate (object obj) {
     printMessage(obj);
   }, "Second Task");

   // use a lambda expression and a named method
   // note that parameters to a lambda don't need
   // to be quoted if there is only one parameter
   Task task3 = new Task((obj) => printMessage(obj), "Third task");

   // use a lambda expression and an anonymous method
   Task task4 = new Task((obj) => {
    printMessage(obj);
   }, "Fourth task");

  task1.Start();
  task2.Start();
  task3.Start();
  task4.Start();

  // wait for input before exiting
  Console.WriteLine("Main method complete. Press enter to finish.");
  Console.ReadLine();
}

 static void printMessage(object message) {
   Console.WriteLine("Message: {0}", message);
  }
 }
}
```

کد بالا را بروش دیگری هم می‌توان نوشت :

```
using System;
using System.Threading.Tasks;

namespace Listing_04 {
class Listing_04 {
  static void Main(string[] args) {
   string[] messages = { "First task", "Second task",
    "Third task", "Fourth task" };

   foreach (string msg in messages) {
     Task myTask = new Task(obj => printMessage((string)obj), msg);
```

```
     myTask.Start();
  }

  // wait for input before exiting
  Console.WriteLine("Main method complete. Press enter to finish.");
  Console.ReadLine();
}

  static void printMessage(string message) {
   Console.WriteLine("Message: {0}", message);
  }
 }
}
```

نکته مهم در کد بالا تبدیل اطلاعات وضعیت Task به رشته کاراکتری است که در عبارت لامبدا مورد استفاده قرار می‌گیرد.
System.Action فقط با داده نوع object کار می‌کند.

خروجی برنامه بالا بصورت زیر است :

```
Main method complete. Press enter to finish.

Message: Second task

Message: Fourth task

Message: First task

Message: Third task
```

البته این خروجی برای شما ممکن است متفاوت باشد چون در سیستم شما ممکن است Taskها با ترتیب متفاوتی اجرا شوند.با کمک Task Scheduler برا حتی می‌توان ترتیب اجرای Taskها را کنترل نمود

نظرات خوانندگان

نویسنده: حسین مرادی نیا
تاریخ: ۱۳۹۱/۰۴/۰۱ ۳:۳۲

در برنامه بالا ابتدا Taskها را Start کرده و سپس کد زیر اجرا می‌شود:

```
Console.WriteLine("Main method complete. Press enter to finish.");
```

سوال من اینه که چرا عبارت Main Method Complete.Press Enter to finish اول از همه در خروجی نمایش داده می‌شود؟!

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۰۱ ۹:۴۴

نوشتن متد Start به این معنا نیست که همین الان باید Start صورت گیرد. بعد Start دوم و بعد مورد سوم و الی آخر. پردازش موازی به همین معنا است و قرار است این موارد به موازات هم اجرا شوند و نه ترتیبی و پشت سر هم .
در یک برنامه کنسول، متد Main یعنی کدهایی که در ترد اصلی برنامه اجرا می‌شوند. زمان اجرای تمام taskهای تعریف شده، با زمان اجرای ترد اصلی برنامه بسیار بسیار نزدیک است اما ممکن است یک تاخیر چند میلی ثانیه‌ای اینجا وجود داشته باشد و آن هم وهله سازی و در صف قرار دادن taskها و اجرای آن‌ها است.
Task در دات نت 4 از thread pool مخصوص CLR استفاده می‌کند که همان thread pool ایی است که توسط متد ThreadPool.QueueUserWorkItem موجود در نگارش‌های قبلی دات نت، مورد استفاده قرار می‌گیرد؛ با این تفاوت که جهت کارکرد با Tasks بهینه سازی شده است (جهت استفاده بهتر از CPUهای چند هسته‌ای).
همچنین باید توجه داشت که استفاده از یک استخر تردها به معنای درصف قرار دادن کارها نیز هست. بنابراین یک زمان بسیار کوتاه جهت در صف قرار دادن کارها و سپس ایجاد تردهای جدید برای اجرای آن‌ها در اینجا باید درنظر گرفت.

یک منبع بسیار عالی برای مباحث پردازش موازی به همراه توضیحات لازم:
http://www.albahari.com/threading/part5.aspx#_Task_Parallelism

نویسنده: حسین مرادی نیا
تاریخ: ۱۳۹۱/۰۴/۰۱ ۱۶:۵۳

مرسی
خیلی مفید بود
اینطور که من فهمیدم CLR همه Taskهای Start شده را جمع آوری کرده و جهت اجرا درون یک صف قرار می‌دهد.
اما شما گفتید که قرار نیست کارها به ترتیب و پشت سر هم اجرا شوند! حال سوال اینجاست که هدف از درون صف قرار دادن Taskها چیست؟! مگر به صورت موازی اجرا نمیشوند؟!

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۰۱ ۱۷:۲۱

برای اینکه CPUها از لحاظ پردازش موازی دارای توانمندی‌های نامحدودی نیستند و لازم است مکانیزم صف وجود داشته باشد و همچنین برنامه شما تنها برنامه‌ای نیست که حق استفاده از توان پردازشی مهیا را دارد.

عنوان: **Parallel Programming در Vb.Net**
نویسنده: میثم ثوامری
تاریخ: ۱۳۹۱/۰۵/۱۹ ۱۲:۱۴
آدرس: www.dotnettips.info
برچسب‌ها: LINQ, Task Programming Language, Parallel Extensions, VB.NET

حقیقتا تا این لحظه تو پروژه ای استفاده نکردم ولی فکر میکنم یادگیری و استفادش ضروری باشه. ظهورش برمیگرده به .net1 با عنوان Threading. اما کار با Threading خیلی مشکله. من که اینطوری فکر میکنم. حالا با اصلاح کلاس Threading و آمده task خیلی بهتر شده.

گام اول:Threading.Tasks را بعنوان namespace اضافه کنید

یک مثال: این loop در نظر بگیرید

```
Private Sub work()
        While True
        End While
End Sub
```

میخوام برا متد بالا یک task تعریف کنم

```
Task.Factory.StartNew(Sub() work())
```

مثال دوم: یک لیست تعریف میکنم و با استفاد از یک loop میخوام اجزا لیستو چاپ کنم.

```
Dim lst As New List(Of String) From {"meysam", ".nettips", "vahidnasirii"}
Parallel.ForEach(lst, Sub(item) Console.WriteLine("name:{0}", item))
```

مثال سوم: میخوام از این تکنیک تو linq استفاده کنیم:

```
Dim no(9) As Integer
   For i As Integer = 0 To no.Length - 1
        no(i) = i
   Next
Dim result As IEnumerable(Of Double) = no.AsParallel.Select(Function(q) Math.Pow(q,
3)).OrderBy(Function(q) q)
   For Each items In result
        Console.WriteLine(items)
   Next
```

موفق باشید.

نظرات خوانندگان

نویسنده: مرتضی
تاریخ: ۱۳۹۱/۰۵/۱۹ ۱۳:۹

سلام
این کد از لحاظ منطقی درسته و جواب می‌ده ولی کاملا اشتباست چون sub رو بی‌دلیل نوشتی

```
Task.Factory.StartNew(Sub() work())
        'نحوه‌ی صحیح نوشتنش

        Task.Factory.StartNew(AddressOf work)
        '---- یا
        Task.Factory.StartNew(Sub()
                                  While True
                                  End While
                              End Sub)
```

نویسنده: میثم ثوامری
تاریخ: ۱۳۹۱/۰۵/۱۹ ۱۹:۳۵

AddressOf در دستور Threading که قدیمی هست استفاده میشه که عمدتا بصورت:

```
Dim t As New Threading.Thread(AddressOf work)
t.Start()
```

متد Work برای این تعریف شده که مفوم کد برسونه.

نویسنده: مرتضی
تاریخ: ۱۳۹۱/۰۵/۱۹ ۱۹:۴۶

سلام میثم جان اشتباه نکن

AddressOf

ربطی به Thread و یا Task نداره

از AddressOf برای ارجاع به Procedure و Functionها استفاده میشه

نویسنده: میثم ثوامری
تاریخ: ۱۳۹۱/۰۵/۲۰ ۱:۱۲

دوست من منظور من این نبود که AddressOf ارتباطی با Threading داره. منظور من این بود که از زمانی که من Parallel Programmimg کار کردم جایی ندیدم از AddressOf تو دستور Task یا Parallel استفاده کنن.از این دستور تو Thread یا BackgroundWorking استفاده میشد که نسبتا تو نسخه‌های قدیمی .net هستن.