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.