

qq_30523227的博客

目录视图

摘要视图

RSS 订阅

个人资料



爆碎天下

关注

发私信



访问：1683次

积分：188

等级：BLOG > 2

排名：千里之外

原创：17篇 转载：3篇

译文：0篇 评论：0条

文章搜索

文章分类

C# (10)

设计模式 (10)

系统安装 (2)

环境 (0)

环境配置。 (4)

WPF (1)

文章存档

2016年11月 (1)

2016年10月 (8)

2016年09月 (11)

阅读排行

Thread.Start()与ThreadPool... (561)

系统重装之VS+Sql2012+IIS... (155)

WPF 当前上下文中不存在名... (134)

SQLserver2012 附加数据库... (108)

C#--设计模式之享元模式 (100)

用户代码未处理 JsonSerializer... (87)

C#--设计模式之外观模式 (71)

编程之道，在于总结，在于一... (63)

JTAG各类接口引脚定义及含义 (52)

CSDN日报20170712——《AI 大行其道，你准备好了吗？》

征文 | 你会为 AI 转型么？

专家问答 | 资深Java工程师带你解读MyBatis

Thread.Start()与ThreadPool.QueueUserWorkItem实现多线程的两种方式的对比

标签：C# 对象 ThreadPool

2016-10-13 11:23

564人阅读

评论(0)

收藏

举报

分类：

C# (9)

版权声明：本文为博主原创文章，未经博主允许不得转载。

好久没有上菜了，这段时间工作忙了点儿，就少了时间来论坛逛了，今天看到一个有关多线程并发处理的文章，经过整理网上的资料，自己做了些总结。

线程池通过为一个应用程序提供一个由系统管理的辅助线程池使您可以更为有效地使用线程。一个线程监视排到线程池的若干个等待操作的状态。当一个等待操作完成时，线程池中的一个辅助线程就会执行对应的回调函数

Thread.Start()，ThreadPool.QueueUserWorkItem都是在实现多线程并行编程时常用的方法。两种方式有何异同点，而又该如何取舍？写一个Demo，分别用两种方式实现。观察各自的现象。代码如下：

```
public void doSomething()
{
    OnBegin(new EventArgs());

    // someone does something here

    var r = new Random();

    int sleepTime = r.Next(3000, 180000);

    Thread.Sleep(900000);

    OnCompleted(new EventArgs());
}
```

```
public void doSomething()
{
    OnBegin(new EventArgs());

    // someone does something here

    var r = new Random();
```

C#--设计模式之组合模式	(38)
C#--设计模式之享元模式	(0)
系统重装之VS+Sql2012+IIS...	(0)
C#--设计模式之桥接模式	(0)
C#--设计模式之适配器模式	(0)
C#--设计模式之原型模式	(0)
C#--设计模式之建造者模式	(0)
C#--设计模式之工厂方法模式	(0)
C#--设计模式之简单工厂模式	(0)
C#--设计模式之单例模式	(0)
C#--BackgroundWorker使...	(0)

推荐文章

- * CSDN日报20170706——《屌丝程序员的逆袭之旅》
- * 探讨后端选型中不同语言及对应的Web框架
- * 细说反射，Java 和 Android 开发者必须跨越的坎
- * 深度学习 | 反向传播与它的直观理解
- * ArcGIS 水文分析实战教程——雨量计算与流量统计
- * 每周荐书：Android、Keras、ES6（评论送书）

```
int sleepTime = r.Next(3000, 180000);

Thread.Sleep(900000);

OnCompleted(new EventArgs());
}

for (var i = 0; i < NUMBER_OF_THREADS; i++)
{
    arrWorkMen[i] = new WorkMan()
    {
        WorkStarted = true,
        InstanceID = startThreadNumber
    };

    arrWorkMen[i].BeginHandler += HandleTaskBegin;
    arrWorkMen[i].CompletedHandler += HandleTaskCompleted;

    startThreadNumber++;
}

for (var i = 0; i < NUMBER_OF_THREADS; i++)
{
    Thread.Sleep(2000);

    ThreadPool.QueueUserWorkItem(o => arrWorkMen[i].doSomething());
}
```

通过观察这两种方式运行的结果，线程创建和回收情况，同样的场景，每2秒钟发起一新的线程，且每一线程均休眠2分钟。Thread.Start()实现下，线程一路最高飙升到71个，然后随着2分钟后休眠线程的结束，线程个数始终在70、71之间徘徊。而ThreadPool.QueueUserWorkItem的实现下，线程个数到达最高73后，始终在72、73之间徘徊。总体来说，做同样的事情。ThreadPool方式产生的线程数略高于Thread.Start()。Thread.Start()产生的线程在完成任务后，很快被系统所回收。而ThreadPool（线程池）方式下，线程在完成工作后会被保留一段时间以备resue。所以，当需求需要大量线程并发工作的时候，不建议使用ThreadPool方式，因为它会保持很多额外的线程。

顶

0

踩

0

- 上一篇

SQLserver2012 附加数据库出错之解决办法
- 下一篇

C#--设计模式之组合模式

相关文章推荐

- [NET][THREAD]C#的多线程机制探索【精】
- C#的多线程（2）——机制探索
- .Net多线程总结(一)
- .Net线程问题解答
- c#多线程
- C#综合揭秘——细说多线程（上）
- 多线程
- C#的多线程机制探索
- C#中利用委托实现多线程跨线程操作
- C# 多线程



家用泳池



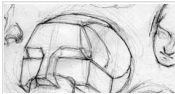
一点点加盟



thread



手绘学习



素描培训班

猜你在找

- 机器学习之概率与统计推断
- 机器学习之数学基础
- 机器学习之凸优化
- 机器学习之矩阵
- 响应式布局全新探索
- 探究Linux的总线、设备、驱动模型
- 深度学习基础与TensorFlow实践
- 深度学习之神经网络原理与实战技巧
- 前端开发在线峰会
- TensorFlow实战进阶：手把手教你做图像识别应用

查看评论

暂无评论

您还没有登录,请[登录]或[注册]

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

