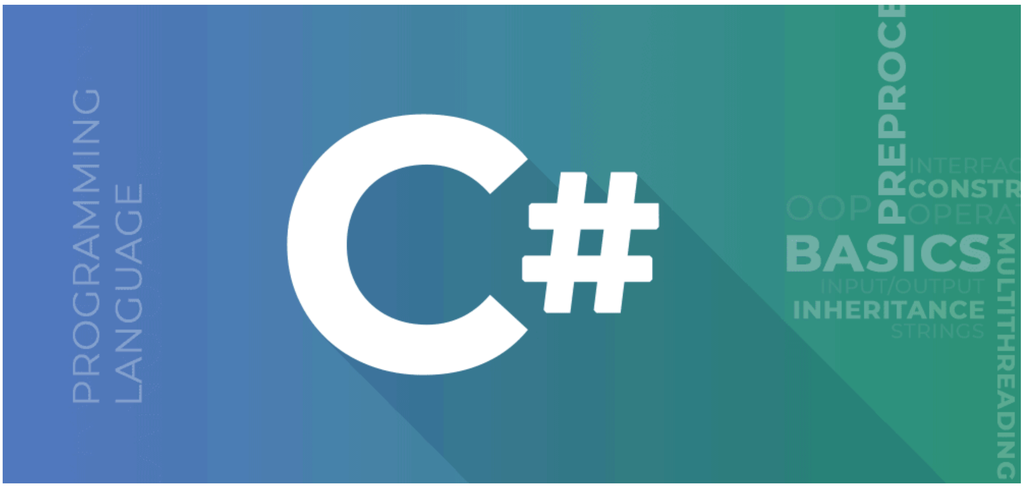
C#异步定时器：精准定期执行异步任务的完整指南

作者：微信公众号：【架构师老卢】 12-28 8:30 242



概述：在C#中，通过System.Threading.Timer或System.Timers.Timer，结合异步方法，实现了简单可靠的定期运行异步任务的机制，为定时任务提供了便捷的解决方案。

在C#中，可以使用`System.Threading.Timer`或`System.Timers.Timer`等定时器类，配合异步方法实现定期运行。这些定时器在指定的时间间隔触发回调函数，从而执行异步操作。

2. 方法说明

2.1 使用`System.Threading.Timer`

```
Timer timer = new Timer(AsyncMethodCallback, null, TimeSpan.Zero, TimeSpan.FromSeconds(5));

async void AsyncMethodCallback(object state)
{
    // 异步操作的内容
}
```

2.2 使用`System.Timers.Timer`

```
System.Timers.Timer timer = new System.Timers.Timer(5000);
timer.Elapsed += async (sender, e) => await AsyncMethod();
timer.Start();
```

3. 步骤说明

3.1 使用`System.Threading.Timer`

1. 创建`Timer`实例，设置初始延迟和时间间隔。
2. 编写异步方法作为定时器的回调函数。
3. 在回调函数中执行异步操作。

3.2 使用`System.Timers.Timer`

1. 创建`System.Timers.Timer`实例，设置时间间隔。
2. 编写异步方法作为定时器的事件处理程序。
3. 在事件处理程序中执行异步操作。

4. 实例源代码

4.1 使用`System.Threading.Timer`

```
using System;
using System.Threading;

class Program
{
    static void Main()
    {
        Timer timer = new Timer(AsyncMethodCallback, null, TimeSpan.Zero, TimeSpan.FromSeconds(5));

        // 防止主线程退出
        Console.ReadLine();
    }

    async static void AsyncMethodCallback(object state)
    {
        Console.WriteLine($"Async method executed at {DateTime.Now}");
        // 异步操作的内容
    }
}
```

4.2 使用`System.Timers.Timer`

```
using System;
using System.Timers;

class Program
{
    static void Main()
    {
        System.Timers.Timer timer = new System.Timers.Timer(5000);
        timer.Elapsed += async (sender, e) => await AsyncMethod();
        timer.Start();

        // 防止主线程退出
        Console.ReadLine();
    }

    async static Task AsyncMethod()
    {
        Console.WriteLine($"Async method executed at {DateTime.Now}");
        // 异步操作的内容
    }
}
```

5. 注意事项及建议

- 注意异步方法的编写和调用，确保异步操作能够正确执行。
- 考虑定时器回调函数的异常处理，以防止未捕获的异常导致程序崩溃。

- 尽量避免在异步回调函数中进行长时间运行的同步操作，以免阻塞定时器线程。

通过使用`System.Threading.Timer`或`System.Timers.Timer`，结合异步方法，我们可以在C#中实现定期运行异步操作的功能。选择合适的定时器类取决于具体需求，而注意异步方法的编写和异常处理则是确保程序稳定运行的关键。