

如何让线程支持超时？

使用 **CancellationTokenSource**

代码

```
1     private static void TimeoutTest1()
2     {
3         var cts = new CancellationTokenSource();
4
5         var thread = new Thread(() =>
6         {
7             Console.WriteLine(String.Format("线程{0}执行中", Thread.CurrentThread.ManagedThreadId));
8             Thread.Sleep(10000);
9             Console.WriteLine(String.Format("线程{0}执行中", Thread.CurrentThread.ManagedThreadId));
10        });
11
12        cts.Token.Register(() =>
13        {
14            thread.Abort();
15        });
16        cts.CancelAfter(1000);
17
18        thread.Start();
19        thread.Join();
20
21        Console.WriteLine(String.Format("线程{0}的状态: {1}", thread.ManagedThreadId, thread.ThreadState));
22    }
```

输出



备注

这里采用了 **Abort** 终止了线程，**CancellationTokenSource** 也支持其它模式，可以去官方看看文档。

使用 **Join**

代码

```
1     private static void TimeoutTest2()
2     {
3         var thread = new Thread(() =>
4         {
5             Console.WriteLine(String.Format("线程{0}执行中", Thread.CurrentThread.ManagedThreadId));
6             Thread.Sleep(10000);
7             Console.WriteLine(String.Format("线程{0}执行中", Thread.CurrentThread.ManagedThreadId));
8         });
9
10        thread.Start();
11        thread.Join(1000);
12        thread.Abort();
13
14        Console.WriteLine(String.Format("线程{0}的状态: {1}", thread.ManagedThreadId, thread.ThreadState));
15    }
```

输出



基于 Task 的实现
代码

```

1      private static void TimeoutTest3()
2      {
3          var cts = new CancellationTokenSource();
4          var task = new Task(() =>
5          {
6              while (true)
7              {
8                  cts.Token.ThrowIfCancellationRequested();
9
10                 Console.WriteLine("xxxxxx");
11                 Thread.Sleep(1000);
12             }
13         }, cts.Token);
14
15         task.Start();
16
17         cts.CancelAfter(5000);
18
19         Console.ReadLine();
20     }

```

输出



如何让线程在执行结束后销毁？

线程执行完、遇到未处理异常和被终止后就自动不可用了，如果是垃圾，自然会被 GC 给回收，有一点需要说明的是：线程的未处理异常会导致应用程序的终止，一个线程的异常不会自动冒泡到其它线程。

备注

我学习多线程知识感觉到的一个好处就是：让我对数据库并发有了更深刻的认识了，找个机会会写写线程的乐观锁和数据库的乐观锁的比较，思路基本一样。