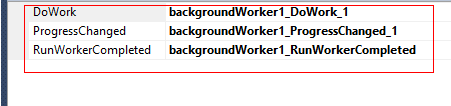


BackgroundWorker 的主要属性

一个是支持报告进度； 一个是支持取消工作； 如果设置为 false, 执行相关操作将导致抛出错误



BackgroundWorker 的主要三个事件：

完整的程序如下：

启动：

if (!this.backgroundWorker1.IsBusy)

{

this.backgroundWorker1.RunWorkerAsync(100); // 重复启动会抛出错误

}

取消：

if (this.backgroundWorker1.WorkerSupportsCancellation)

{

this.backgroundWorker1.CancelAsync();// 不支持取消会抛出错误， 重复取消不会抛出错误

}

工作线程：

private void backgroundWorker1\_DoWork\_1(object sender, DoWorkEventArgs e)

{

BackgroundWorker bgwrk = sender as BackgroundWorker;

int i = 0;

int sum = 0;

while (i <= (int)e.Argument)

{

// 以下代码是取消操作的关键，没有这些代码， 调用取消也是白费，线程继续执行直到完成

if (bgwrk.CancellationPending)

{

e.Result = sum;

// 此属性很重要，如果不设置，即使取消也当作是正常完成

// 如果不设置，默认为 False, 那么在完成事件里，e.Cancelled = false; e.Result 的值可以访问

// 如果设置为 True, 那么在完成事件里， e.Cancelled = true; e.Result 的值不能访问，会抛出错误

e.Cancel = true;

break;

}

sum += i++;

if(bgwrk.WorkerReportsProgress) bgwrk.ReportProgress(i \* 100 / (int)e.Argument);

this.listBox1.Items.Add(string.Format("Working:{0} - {1}", i, sum));

Thread.Sleep(200);

}

e.Result = sum;

}

进度线程：

private void backgroundWorker1\_ProgressChanged\_1(object sender, ProgressChangedEventArgs e)

{

this.progressBar1.Value = e.ProgressPercentage;

this.listBox1.Items.Add(string.Format("Progress:{0}", e.ProgressPercentage));

}

完成线程：

private void backgroundWorker1\_RunWorkerCompleted(object sender, RunWorkerCompletedEventArgs e)

{

BackgroundWorker bgwrk = sender as BackgroundWorker;

if (e.Error!=null)

{

MessageBox.Show(string.Format("Complete with Error:{0}-{1}",

e.Error.Message,

e.Error.StackTrace));

}

if (e.Cancelled)

{

// 不能访问 e.Result, 会抛出错误

MessageBox.Show(string.Format("Cancelled:{0}", bgwrk.IsBusy));

}

else

{

// 可以访问 e.Result

MessageBox.Show("Done:" + e.Result.ToString());

}

}