MouStudio 技术储备之　C#进度条模态窗体

模态进度条窗体实现

实现原理

窗体设计

程序设计

执行效果

功能加强版进度条模态窗体

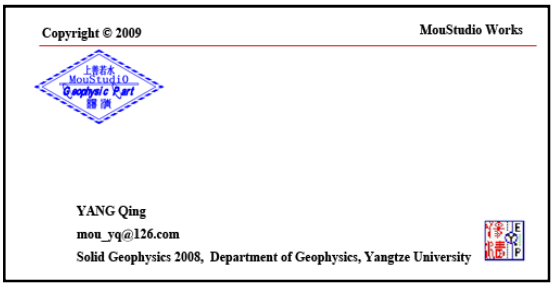
进度条控件的优化

控件制作复用升级

3 Appendix

中子线程控制进度条的一个简单例子

本文简单程序代码图片



MouStudio 技术储备之　C#进度条模态窗体

【介绍】

以前一直就想实现这个功能，结果找了半天资料都没有能够找到符合自己要求的，总体上大概有以下几种形式的进度条窗体显示：

（ ）　玩些假的进度显示，进度条一个劲的显示，进度值到头又重新来，直到主程序将其关闭；

（ ）　可以实现实时的进度更新，但是由于基于的是单进程的控制，在进度条更新的过程中会出现程序假死的现象；

（ ）　更有时基于一般窗体的实现，但是这样就不能实现对主程序的锁定了，但有时候也是需要这个功能的。

本文介绍了一种基于模态对话框的进度条显示窗体，可以很实时准确的显示进度值，并且给予多线程来操作，保证程序的“活性”。此功能的实现基于参考文献：_____。并在此基础上增加了进度条窗体上的功能。

模态进度条窗体实现

实现原理

这个问题会让一些初学者感到困惑，一方面模态窗体在打开之后(Form.ShowDialog).主线程无法继续执行下面的事务；另一方面，又要在处理事务的时候来同步这个窗体里面的进度条。这两件事情都必须做，却都不能先做

所以很多人不得不把事务处理写到模态窗体当中去，这样可以达到目的，却造成了代码结构混乱，而且子窗体无法复用，如果有多个事物，不得不为每个事务来写控制窗体。

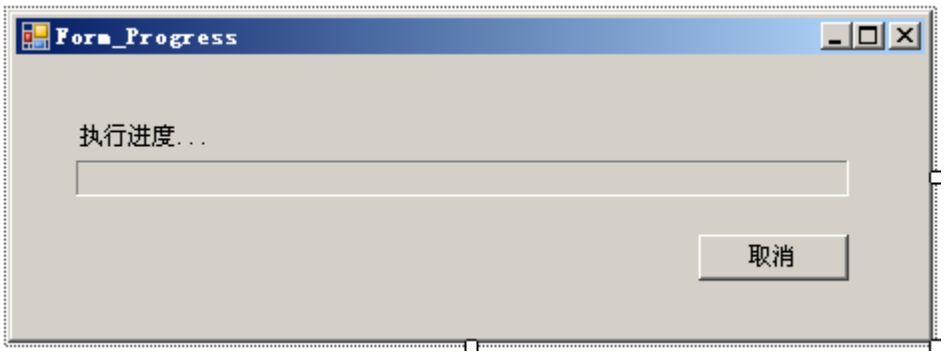
这里我们介绍一种比较直接了当的方法，就是主线程中开启两个子线程，一个用于显示模态窗体，一个用于处理事务，　并同时更新第一个线程中的窗体信息。

这里的知识范畴会包含线程的基础用法，线程和 UI 之间的交互，以及委托的基本用法。如果你还不了解这几点，可以参考一些其他资料，比如我先前写过的，在同一窗体中用子线程来控制进度条：

窗体设计

这里的应用稍微复杂一些，你可以先用 VS2008 新建一个 Windows Application，命名为“进度条模态子窗体”，随后将窗体命名为 Form_Main，并放置一个 Button 作为“进度条测试”用；如下图所示：

接着增加一个进度条的显示子窗体，命名为“Form_Progress”，并放置一个 Button 作为“取消”进度显示操作，再放一个进度条控件“ProgressBar1”。如下图所示：



程序设计

窗体设计完成后，接下来开始编码工作。

首先是在在主窗体中：Form_Main.cs

```
using System
using System Collections Generic
using System ComponentModel
using System Data
using System Drawing
using System Linq
using System Text
using System Windows Forms

namespace 进度条模态子窗体
    public partial class Form_Main   Form
        public Form_Main
            InitializeComponent


        delegate void dShowForm
```

```
        Form_Progress frm_Progress   new Form_Progress

          显示窗体
        void ShowForm
        
          if  this InvokeRequired
            this Invoke new dShowForm this ShowForm
          else
            frm_Progress ShowDialog this
        

          控制进度
        void SetProgress
        
          for  int i      i         i
            if  frm_Progress DialogResult    DialogResult Cancel
                判断取消
              break
            else
                模拟进度
              frm_Progress SetProgress      i
              System Threading Thread Sleep
        

          测试开始
        private void btn_Test_Click object sender  EventArgs e
          new System Threading Thread new
System Threading ThreadStart ShowForm   Start
          new System Threading Thread new
System Threading ThreadStart SetProgress   Start
```

然后是进度条窗体 Form_Progress.cs

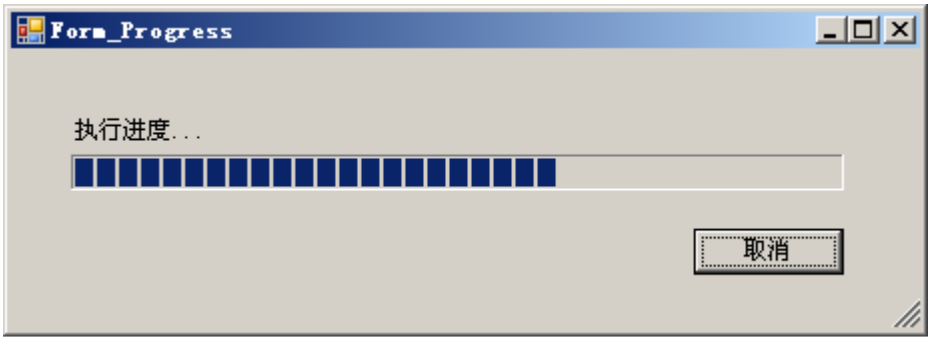
```
using System
using System Collections Generic
using System ComponentModel
using System Data
using System Drawing
using System Linq
using System Text
using System Windows Forms

namespace 进度条模态子窗体
    public partial class Form_Progress   Form
        public Form_Progress
            InitializeComponent


        public delegate void dSetProgress int total  int current

        public void SetProgress int total  int current

            if  this InvokeRequired

                try

                    this Invoke new dSetProgress this SetProgress
                        new object     total  current

                catch

            else

                this progressBar1 Maximum   total
                this progressBar1 Value   current

                //YQ Test
                  达到最大值后退出
                if  this progressBar1 Value
this progressBar1 Maximum

                    this DialogResult   DialogResult Cancel
```

```
        取消操作
    private void btn_Cancel_Click object sender  EventArgs e
        this DialogResult   DialogResult Cancel
```

执行效果

主程序运行界面如下：

进度条窗体界面如下：



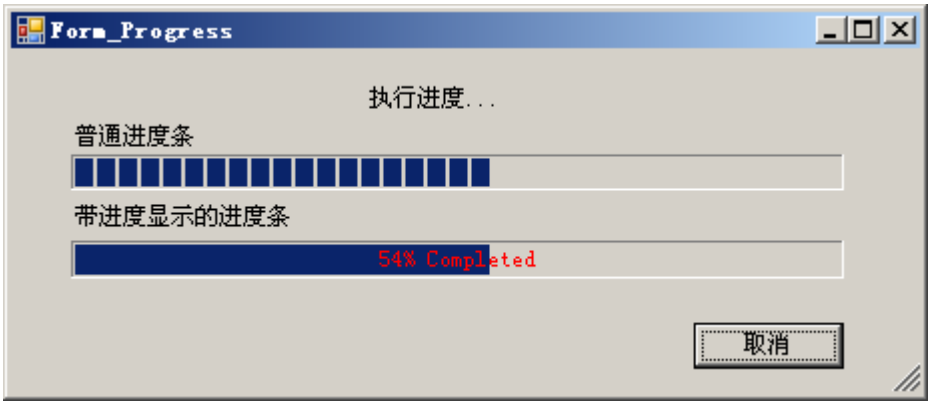
功能加强版进度条模态窗体

针对原来已经实现的功能，为了能够在今后进一步的实现程序的复用，将对进度条模态子窗体的 GUI 方面进行优化。

进度条控件的优化

关于进度条的显示，一般的进度条控件只能显示进度的滚动条，现在需要对其增加完成进度的实时显示，简单点的方法这样其实可以通过拖放一个别人已经完成的控件来实现：
窗体设计如下：

程序执行效果如下：



控件制作复用升级

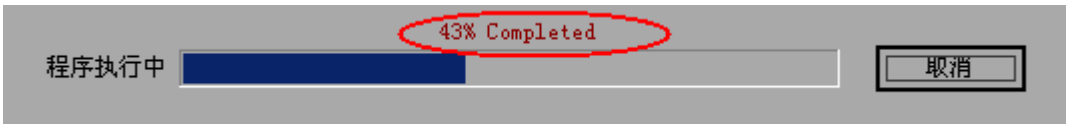
为了能够进一步的使用进度条模态窗体功能，在此将程序制作成.NET 组件，总共分为两类控件：（ ）像上面阐述的以线程控制，并实时显示进度值的进度条；（ ）只是显示功能的圆形状态表示进度条。

控件总类程序

```cs
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

using System.Threading;

namespace LibProgressForm {
    public class Class_ProgressManage {

        //------- 父窗体 -------
        private Form frm_Parent = null;
        public Form Progress_Form_Parent
        {
            get { return frm_Parent; }
            set { frm_Parent = value;
            }
        }

        //------- 进度条窗体类型 -------
        private ProgressFormStyle style_Progress = ProgressFormStyle.Circle_Waiting;
        public ProgressFormStyle ProgressStyle
        {
            get {return style_Progress;}
            set{
                style_Progress = value;
            }
        }

        //------- 进度条的最大值 -------
        private int iNum_ProgressMax = 100;
        public int ProgressMax
        {
            get {
                iNum_ProgressMax = frm_Progress_Normal.progressBar1.Maximum;
                return iNum_ProgressMax;
            }
            set {
                iNum_ProgressMax = value;
                frm_Progress_Normal.progressBar1.Maximum = iNum_ProgressMax;
            }
        }

        //------- 进度条的最小值 -------
        private int iNum_ProgressMin = 0;
        public int ProgressMin {
            get {
                iNum_ProgressMin = frm_Progress_Normal.progressBar1.Minimum;
                return iNum_ProgressMin;
            }
            set {
                iNum_ProgressMin = value;
                frm_Progress_Normal.progressBar1.Minimum = iNum_ProgressMin;
            }
        }

        //------- 进度条的当前值 -------
        private int iNum_ProgressValue = 0;
        public int ProgressValue {
            get {
                iNum_ProgressValue = frm_Progress_Normal.progressBar1.Value;
                return iNum_ProgressValue;
            }
            set {
                iNum_ProgressValue = value;
                frm_Progress_Normal.SetProgress(iNum_ProgressMax, iNum_ProgressValue);
            }
        }




        //------- 进度条窗体定义 -------
        //普通进度条窗体
        private Form_Progress_Normal frm_Progress_Normal = new Form_Progress_Normal();
        //圆形等待窗体，mac
        private Form_Waiting_Circle frm_Waiting_Circle = new Form_Waiting_Circle();

        //代理定义
        private delegate void dShowForm();
        //显示窗体==========================================================
        private void ShowNormalProgressForm() {
            if (frm_Parent == null)
            {
                MessageBox.Show("请指定使用此控件的父窗体");
                return;
            }
            if (frm_Parent.InvokeRequired) {
                frm_Parent.Invoke(new dShowForm(this.ShowNormalProgressForm));
            }
            else {
                frm_Progress_Normal.ShowDialog(frm_Parent);
            }
        }

        private void ShowWaitCircleForm(){
            if (frm_Parent == null)
            {
                MessageBox.Show("请指定使用此控件的父窗体");
                return;
            }
            if (frm_Parent.InvokeRequired) {
                frm_Parent.Invoke(new dShowForm(this.ShowWaitCircleForm));
            }
```

实时进度条控件

开发效果：



程序代码：

```
using System
using System Collections Generic
using System ComponentModel
using System Data
using System Drawing
using System Linq
using System Text
using System Windows Forms

namespace LibProgressForm
    public partial class Form_Progress_Normal    Form
        public Form_Progress_Normal
            InitializeComponent


        public delegate void dSetProgress int total  int current

        public void SetProgress int total  int current
            if  this InvokeRequired
                try
                    this Invoke new dSetProgress this SetProgress
                        new object     total  current
                catch
            else
                this progressBar1 Maximum   total
                this progressBar1 Value   current

                //YQ Test
                  达到最大值后退出
                if  this progressBar1 Value
this progressBar1 Maximum
                    this DialogResult   DialogResult Cancel
                    this Close
```

```
                UpdateText
                Invalidate
```

```
          取消
        private void btn_Cancel_Click object sender  EventArgs e
            this DialogResult   DialogResult Cancel
            this Close
```

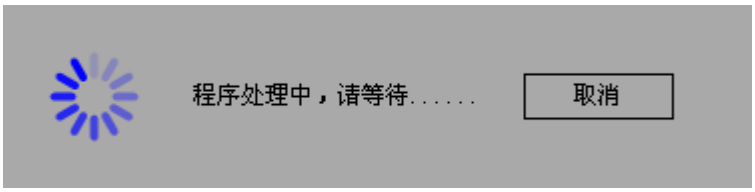
```
          设置显示进度条的信息

        bool ShowPercentage   true
        string CenterText   null
        private void progressBar1_LocationChanged object sender
EventArgs e
            //UpdateText();
```

```
        private void UpdateText
            string s
            if  ShowPercentage
                int percent    int    double  progressBar1 Value
progressBar1 Minimum
                    double  progressBar1 Maximum
progressBar1 Minimum
                s   percent ToString     "% Completed"
            else
                if  string IsNullOrEmpty CenterText
                    //Dont draw anything
                    return
                else
                    s   CenterText
            lb_Message Text   s
```

```cs
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace LibProgressForm {
    public partial class Form_Progress_Normal : Form {
        public Form_Progress_Normal() {
            InitializeComponent();
        }

        public delegate void dSetProgress(int total, int current);

        public void SetProgress(int total, int current) {
            if (this.InvokeRequired) {
                try {
                    this.Invoke(new dSetProgress(this.SetProgress),
                        new object[] { total, current });
                }
                catch { }
            }
            else {
                this.progressBar1.Maximum = total;
                this.progressBar1.Value = current;

                //YQ Test
                //达到最大值后退出
                if (this.progressBar1.Value == this.progressBar1.Maximum) {
                    this.DialogResult = DialogResult.Cancel;
                    this.Close();
                }

                UpdateText();
                Invalidate();
            }
        }

        //取消
        private void btn_Cancel_Click(object sender, EventArgs e) {
            this.DialogResult = DialogResult.Cancel;
            this.Close();
        }

        //设置显示进度条的信息=============================================
        bool ShowPercentage = true;
        string CenterText = null;
        private void progressBar1_LocationChanged(object sender, EventArgs e) {
            //UpdateText();
        }

        private void UpdateText() {
            string s;
            if (ShowPercentage) {
                int percent = (int)(((double)(progressBar1.Value - progressBar1.Minimum)
                    / (double)(progressBar1.Maximum - progressBar1.Minimum)) * 100);
                s = percent.ToString() + "% Completed";
            }
            else {
                if (string.IsNullOrEmpty(CenterText)) {
                    //Dont draw anything
                    return;
                }
                else {
                    s = CenterText;
                }
            }
            lb_Message.Text = s;
//              using (Graphics gr = progressBar1.CreateGraphics()) {
//                  gr.DrawString(s, Font, new SolidBrush(ForeColor),
//                      new PointF(Width / 2 - (gr.MeasureString(s, Font).Width / 2.0F),
//                          Height / 2 - (gr.MeasureString(s, Font).Height / 2.0F)));
//              }
        }

        private void progressBar1_ParentChanged(object sender, EventArgs e) {
            //UpdateText();
        }

        private void Form_Progress_Normal_Paint(object sender, PaintEventArgs e) {
            //UpdateText();
        }


    }
}

```

状态进度条控件

开发效果：



程序代码：

```
using System
using System Collections Generic
using System ComponentModel
using System Data
using System Drawing
using System Linq
using System Text
using System Windows Forms

namespace LibProgressForm
    public partial class Form_Waiting_Circle    Form
        public Form_Waiting_Circle
            InitializeComponent


        private void btn_Cancel_Click object sender  EventArgs e
            circularProgressControl1 Stop
            this DialogResult    DialogResult Cancel
            this Close


        private void Form_Waiting_Circle_Load object sender  EventArgs
e
            circularProgressControl1 Start
```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace LibProgressForm {
    public partial class Form_Waiting_Circle : Form {
        public Form_Waiting_Circle() {
            InitializeComponent();
        }

        private void btn_Cancel_Click(object sender, EventArgs e) {
            circularProgressControl1.Stop();
            this.DialogResult = DialogResult.Cancel;
            this.Close();
        }

        private void Form_Waiting_Circle_Load(object sender, EventArgs e) {
            circularProgressControl1.Start();
        }
    }
}
```

控件使用举例

运行效果

程序代码：

```
using System
using System Collections Generic
using System ComponentModel
using System Data
using System Drawing
using System Linq
using System Text
using System Windows Forms

using LibProgressForm      引用自定义的进度条模态窗体控件
using System Threading
```

```
namespace ProgressFormTest
    public partial class Form1   Form
        public Form1
            InitializeComponent
            waitCircle Progress_Form_Parent   this
            waitCircle ProgressStyle
ProgressFormStyle Circle_Waiting

            progress Progress_Form_Parent   this
            progress ProgressStyle
ProgressFormStyle Normal_Progress


          状态进度条
        private Class_ProgressManage waitCircle   new
Class_ProgressManage
          实时进度条
        private Class_ProgressManage progress   new
Class_ProgressManage

        private void btn_Test_Click object sender  EventArgs e
            progress showDialog
              采用新线程启动进度控制
            new System Threading Thread new
System Threading ThreadStart SetProgress2   Start


          控制进度
        void SetProgress2
            int iMax
            progress ProgressMax   iMax
            for  int i      i    iMax  i
                progress ProgressValue   i
                System Threading Thread Sleep


        private void btn_TestWaiting_Click object sender  EventArgs e

            //Class_ProgressManage progress
            waitCircle showDialog


```

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

using LibProgressForm; //引用自定义的进度条模态窗体控件
using System.Threading;

namespace ProgressFormTest {
    public partial class Form1 : Form {
        public Form1() {
            InitializeComponent();
            waitCircle.Progress_Form_Parent = this;
            waitCircle.ProgressStyle = ProgressFormStyle.Circle_Waiting;

            progress.Progress_Form_Parent = this;
            progress.ProgressStyle = ProgressFormStyle.Normal_Progress;

        }

        //状态进度条
        private Class_ProgressManage waitCircle = new Class_ProgressManage();
        //实时进度条
        private Class_ProgressManage progress = new Class_ProgressManage();

        private void btn_Test_Click(object sender, EventArgs e) {
            progress.showDialog();
            //采用新线程启动进度控制
            new System.Threading.Thread(new System.Threading.ThreadStart(SetProgress2)).Start();
        }

        //控制进度
        void SetProgress2() {
            int iMax = 100;
            progress.ProgressMax = iMax;
            for (int i = 0; i <= iMax; i++) {

                progress.ProgressValue = i;

                System.Threading.Thread.Sleep(50);
            }
        }

        private void btn_TestWaiting_Click(object sender, EventArgs e) {
            //Class_ProgressManage progress
            waitCircle.showDialog();
        }


    }
}

```

3 Appendix

c#中子线程控制进度条的一个简单例子

这个问题来自社区提问，代码保留一份用来以后回答

```
using System;
using System.ComponentModel;
using System.Windows.Forms;
namespace WindowsApplication4
{
    /// <summary>
    /// gui 类
    /// </summary>
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void button1_Click( object sender, EventArgs e)
        {
            // 用子线程工作
            new System.Threading.Thread( new System.Threading.ThreadStart(StartDownload)).Start();
        }
        // 开始下载
        public void StartDownload()
        {
            Downloader downloader = new Downloader();
            downloader.onDownLoadProgress += new Downloader.dDownloadProgress(downloader_onDownLoadProgress);
            downloader.Start();
        }
        // 同步更新 ui
        void downloader_onDownLoadProgress( long total, long current)
        {
            if ( this .InvokeRequired)
            {
                this .Invoke( new Downloader.dDownloadProgress(downloader_onDownLoadProgress), new object [] { total, current } );
```

```
                }
                else
                {
                    this .progressBar1.Maximum  =  ( int )total;
                    this .progressBar1.Value  =  ( int )current;
                }
            }
        }

        /**/ ///  <summary>
        ///  下载类
        ///  </summary>
        public  class  Downloader
        {
            // 委托
            public  delegate  void  dDownloadProgress( long  total, long  current);
            // 事件
            public  event  dDownloadProgress onDownLoadProgress;
            // 开始模拟工作
            public  void  Start()
            {
                for  ( int  i  =  0 ; i  <  100 ; i ++ )
                {
                    if  (onDownLoadProgress  !=  null )
                        onDownLoadProgress( 100 , i);
                    System.Threading.Thread.Sleep( 100 );
                }
            }
        }
    }
```

本文简单程序代码图片

主窗体代码：

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace 进度条模态子窗体 {
    public partial class Form_Main : Form {
        public Form_Main() {
            InitializeComponent();
        }

        delegate void dShowForm();
        Form_Progress frm_Progress = new Form_Progress();

        //显示窗体
        void ShowForm()
        {
            if (this.InvokeRequired)
            {
                this.Invoke(new dShowForm(this.ShowForm));
            }
            else
            {
                frm_Progress.ShowDialog(this);
            }
        }

        //控制进度
        void SetProgress()
        {
            for (int i = 1; i <= 100; i++)
            {
                if (frm_Progress.DialogResult == DialogResult.Cancel)
                {
                    //判断取消
                    break;
                }
                else
                {
                    //模拟进度
                    frm_Progress.SetProgress(100, i);
                    System.Threading.Thread.Sleep(50);
                }
            }
        }

        //测试开始
        private void btn_Test_Click(object sender, EventArgs e) {
            new System.Threading.Thread(new System.Threading.ThreadStart(ShowForm)).Start();
            new System.Threading.Thread(new System.Threading.ThreadStart(SetProgress)).Start();
        }
    }
}
```

进度条子窗体代码：

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace 进度条模态子窗体 {
    public partial class Form_Progress : Form {
        public Form_Progress() {
            InitializeComponent();
        }

        public delegate void dSetProgress(int total, int current);

        public void SetProgress(int total, int current)
        {
            if (this.InvokeRequired)
            {
                try
                {
                    this.Invoke(new dSetProgress(this.SetProgress),
                        new object[] { total, current });
                }
                catch { }
            }
            else
            {
                this.progressBar1.Maximum = total;
                this.progressBar1.Value = current;

                //YQ Test
                //达到最大值后退出
                if (this.progressBar1.Value == this.progressBar1.Maximum)
                {
                    this.DialogResult = DialogResult.Cancel;
                }
            }
        }

        //取消操作
        private void btn_Cancel_Click(object sender, EventArgs e) {
            this.DialogResult = DialogResult.Cancel;
        }
    }
}

```