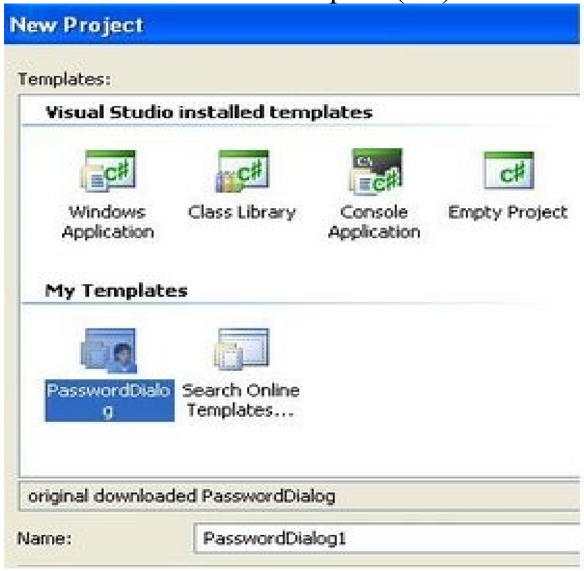
Starting With C# in Visual Studio

When one starts up VS(C#) and invokes the "New Project" menu item, one sees:



original downloaded PasswordDialog

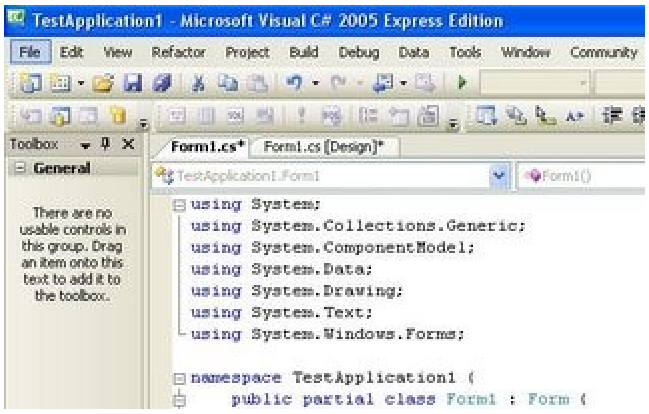
Name: PasswordDialog1

and when one choose a Windows Application and gives it a unique name, one gets a window showing a form. Double clicking on this form shows the form's code.
On my machine, that code is:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace TestApplication1 {
      public partial class Form1 : Form {
      public Form1() {
       InitializeComponent();
       }
      private void Form1_Load(object sender, EventArgs e) {
      }
   }
}
```

since I named my application "TestApplication1". Notice that the figure (above) shows a template highlighted, but when I created the above code, I clicked on the upper left icon.

The code (above) "looks" like this on the screen:

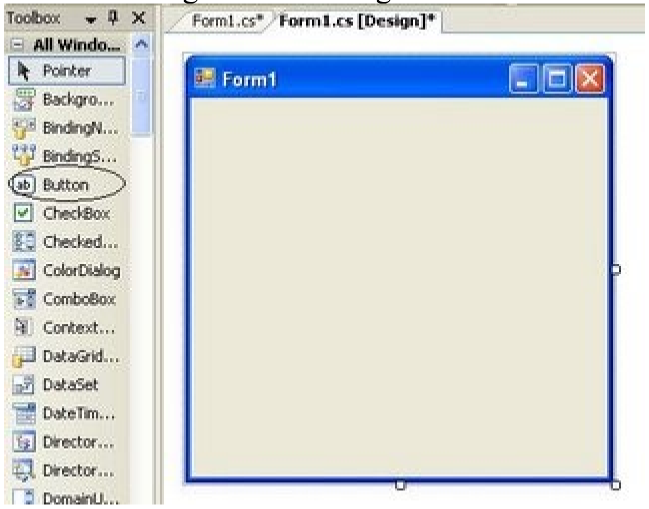

TestApplication1 - Microsoft Visual C# 2005 Express Edition
File Edit View Refactor Project Build Debug Data Tools Window Community
Toolbox
General
Form1.cs*
Form1.cs [Design]*
TestApplication1.Form1
Form1()

There are no usable controls in this group. Drag an item onto this text to add it to the toolbox.

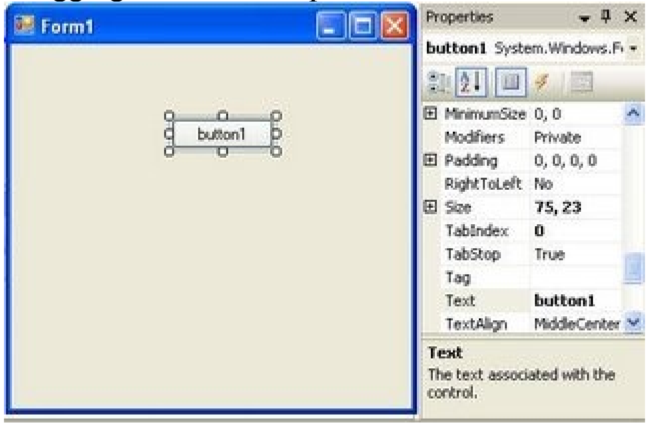
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

namespace TestApplication1 {
    public partial class Form1 : Form {
```

There are two tabs on the top, "Form1.cs*" and "Form1.cs[design]*". Clicking on one of these two tabs switches us from the code to the Design and *vice versa*.

Making the form design active makes the screen look like:



where the "Toolbox" in the upper left corner has an item circled (by me) where one can click next (to make active).
Dragging it to the form places a button on the form:



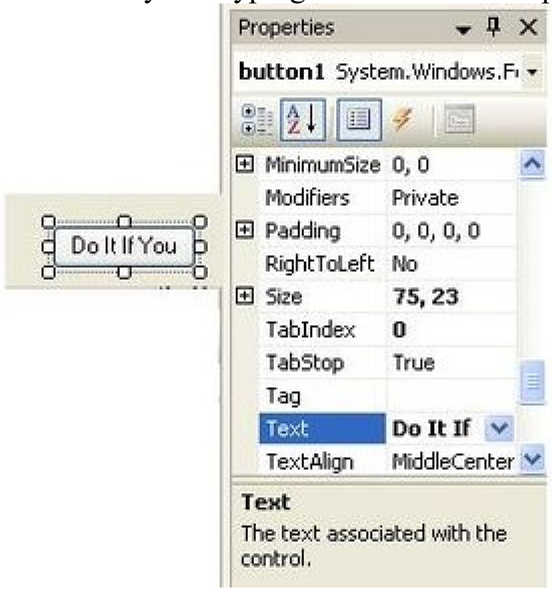
MinimumSize	0, 0
Modifiers	Private
Padding	0, 0, 0, 0
RightToLeft	No
Size	75, 23
TabIndex	0
TabStop	True
Tag	
Text	button1
TextAlign	MiddleCenter

Text
The text associated with the control.

where we've juxtaposed two screen parts which will be separated on your screen.

You will see that what we've got is a button and "button1" written on its face. On the Properties window, you will see "Text" and to the right of that, "button1". You can change "button1" to "Do It If

You Dare" by overtyping the text in the Properties Text value box. Try it, you'll like it:

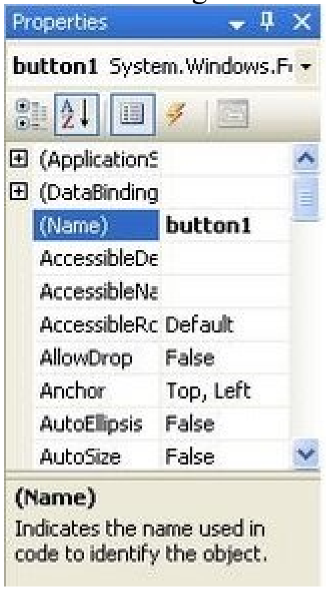


⊞ MinimumSize	0, 0
Modifiers	Private
⊞ Padding	0, 0, 0, 0
RightToLeft	No
⊞ Size	**75, 23**
TabIndex	**0**
TabStop	True
Tag	
Text	**Do It If**
TextAlign	MiddleCenter

Text
The text associated with the control.

Wow. Since the button as created by VS(C#) doesn't show all the letters in "Do It If You Dare" resize it using the mouse.

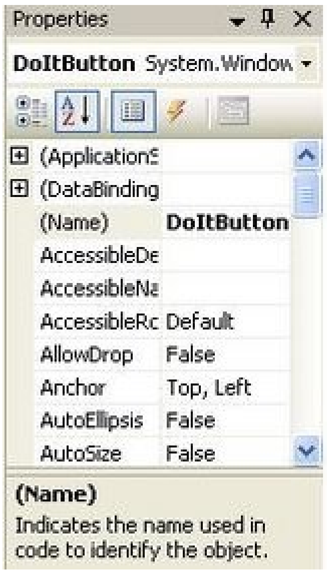
Looking near the top of the Properties list, you will see:



⊞ (ApplicationS	
⊞ (DataBinding	
(Name)	**button1**
AccessibleDe	
AccessibleNa	
AccessibleRo	Default
AllowDrop	False
Anchor	Top, Left
AutoEllipsis	False
AutoSize	False

(Name)
Indicates the name used in code to identify the object.

and you can change the name from button1, which is in my world, meaningless to something like "DoItButton" which will change all the code which refers to this button into a self-referential form tagged with "DoItButton".



(Name)	DoItButton
AccessibleDe	
AccessibleNa	
AccessibleRc	Default
AllowDrop	False
Anchor	Top, Left
AutoEllipsis	False
AutoSize	False

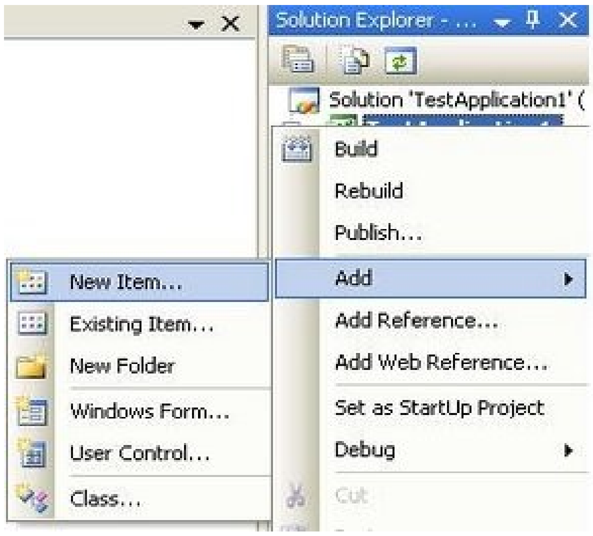
(Name)
Indicates the name used in code to identify the object.

When one double clicks on this button now, an EventHandler will be created for it. This is the place where we put our code for what to do when the button is pushed. The code created is:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace TestApplication1 {
public partial class Form1 : Form {
public Form1() {
    InitializeComponent();
    }
private void Form1_Load(object sender, EventArgs e) {
    }
private void DoItButton_Click(object sender, EventArgs e) {
    }
}
}
```
where the line “private void DoIt...” would have read "button1_Click” had we not re-named the button.

Our interest is in setting up an elementary application which will not only act as a kind of template for future developmental work, but also illustrate how that developmental work can be incorporated into future products efficiently. Thus, we are going to create another form, and in that (daughter) form, we are going to do our actual experimental coding, passing an answer back to the above (main) application. We are therefore going to add a new class.

To do this, right click on the "TestApplication1" (highlighted partially in the accompanying figure) line in Solution Explorer, go down to "Add"



and choose "New Item". Inside this choice, you will see lots of choices, and we choose a new class, which we call "TestApplicationDlg":

```
using System;
using System.Collections.Generic;
using System.Text;
namespace TestApplication1 {
class TestApplicationDlg {
    }
}
```

since this will be a Dialog. Change the appropriate line to:

```
class TestApplicationDlg:System.Windows.Forms.Form
```

and now, when you right click on the C# code, you will be allowed to create a new Form! Incredible!

Here is the new Form's code:

```
using System;
using System.Collections.Generic;
using System.Text;
namespace TestApplication1 {
  class TestApplicationDlg : System.Windows.Forms.Form {
      private System.Windows.Forms.TextBox myTextBox;
      private System.Windows.Forms.Button Submit;
      // added by cwd
      public TestApplicationDlg() {
          InitializeComponent();
          }
// end of addition by cwd
      private void InitializeComponent() {
          this.myTextBox = new System.Windows.Forms.TextBox();
          this.Submit = new System.Windows.Forms.Button();
          this.SuspendLayout();
          //
          // myTextBox
          //
          this.myTextBox.Location = new System.Drawing.Point(73, 55);
          this.myTextBox.Name = "myTextBox";
          this.myTextBox.Size = new System.Drawing.Size(100, 20);
```

```
            this.myTextBox.TabIndex = 0;
            this.myTextBox.Text = "Enter Text Here";
            //
            // Submit
            //
            this.Submit.Location = new System.Drawing.Point(192, 215);
            this.Submit.Name = "Submit";
            this.Submit.Size = new System.Drawing.Size(75, 23);
            this.Submit.TabIndex = 1;
            this.Submit.Text = "Submit";
            this.Submit.UseVisualStyleBackColor = true;
            this.Submit.Click += new System.EventHandler(this.Submit_Click);
            //
            // TestApplicationDlg
            //
            this.ClientSize = new System.Drawing.Size(292, 266);
            this.Controls.Add(this.Submit);
            this.Controls.Add(this.myTextBox);
            this.Name = "TestApplicationDlg";
            this.ResumeLayout(false);
            this.PerformLayout();
            }
//beginning of new code by cwd
      private void Submit_Click(object sender, EventArgs e) {
            DialogResult = System.Windows.Forms.DialogResult.OK;
            Close();
            }
      public String getText() {
            return myTextBox.Text;
            }
      }
  }
```

and here is the slightly adjusted (and to be explained) Form1 code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace TestApplication1 {
  public partial class Form1 : Form {
      public Form1() {
            InitializeComponent();
            }
      private void Form1_Load(object sender, EventArgs e) {
            }
      private void DoItButton_Click(object sender, EventArgs e) {
            TestApplicationDlg myTestApplicationDlg = new TestApplicationDlg();
            if (myTestApplicationDlg.ShowDialog(this) == DialogResult.OK) {
                  String returned_text = myTestApplicationDlg.getText();
                  MessageBox.Show("results = "+returned_text);
                  }
            myTestApplicationDlg.Dispose();
            }
      }
 }
```

The MessageBox.Show gives the results of the Dialog we've initiated above. This set of two programs, created as we've done, acts as a scaffold on which test items can be added to the Dialog and checked so that, when all is ready, that Dialog can be employed elsewhere without copying code. In another note in this set, using References to access the class developed here, will be discussed.

Carl David

Department of Chemistry

University of Connecticut

Storrs, Connecticut 06269-3060

Carl.David@uconn.edu