Windows Forms and the **User Interface**

By Pakita Shamoi, Spring 2013

Introduction

- Windows Forms are the basic building block for most applications
  - They can be configured to provide a variety of user interface (UI) options.
  - They can contain various types of controls
- The developer can create forms of various sizes and shapes and customize them to the user's needs.

Some definitions…

- ***Forms*** are hosts for controls and menu, which provide the main functionality of the UI.
  - ***Forms*** can receive user input in the form of keystrokes or mouse interactions and can display data to the user through hosted controls.
  - You can add forms at design time and at run time as well
    - You can also create new instances of forms at run time by declaring a variable of a type of form and creating an instance of that form.

- There are also special controls called ***container controls*** that are used to control the layout of the UI.

Tuning a form

- The visual appearance of ***UI*** is an important part of your application.
  - A ***UI*** that is poorly designed is difficult to learn and will increase training time and expense.

- You can modify the appearance of your UI by using Windows Forms **properties**
  - **Properties** allow you to customize the look and feel of the form.

- ***Properties :*** Cursor, Font, FormBorderStyle (None, Sizable…), Icon, Opacity (default: 100%), StartPosition, WindowState (e.g. Maximized), Width, Height, TopMost, ControlBox, BackColor,…

Basic modifications on form

```
Form1.Text = "This is Form 1";
```

- The border style of a form determines how the border of the form looks and how it behaves at run time.
  - whether a form is resizable by the user at run time
  - whether various control boxes appear
- None, FixedSingle, Fixed3D, Sizable, FixedToolWindow, …

```
aForm.FormBorderStyle = FormBorderStyle.Fixed3D;
```

- Some properties, such as the *Font* or *Size* *properties, are more complex.*
  - PropertyY = new Class(value,value);

```
aForm.Size = new Size(300,200);
```

Basic modifications on form

- The ***WindowState*** property determines what state the form is in when it first opens.
  - 3 possible values:
    - Normal
    - Minimized
    - Maximized

- Form startup location is influenced by 2 properties:
  - ***StartPosition*** (Manual, CenterScreen, etc.)
  - ***Location***

- You can set a form to always be on top of the UI by setting the ***TopMost*** property to True

Basic modifications on form

- The ***Opacity*** property sets the transparency of the form.
  - The opacity value can range from 0% to 100% (0 to 1), indicating the degree of opacity.
  - 0 represents the complete transparency and 1 represents complete opacity

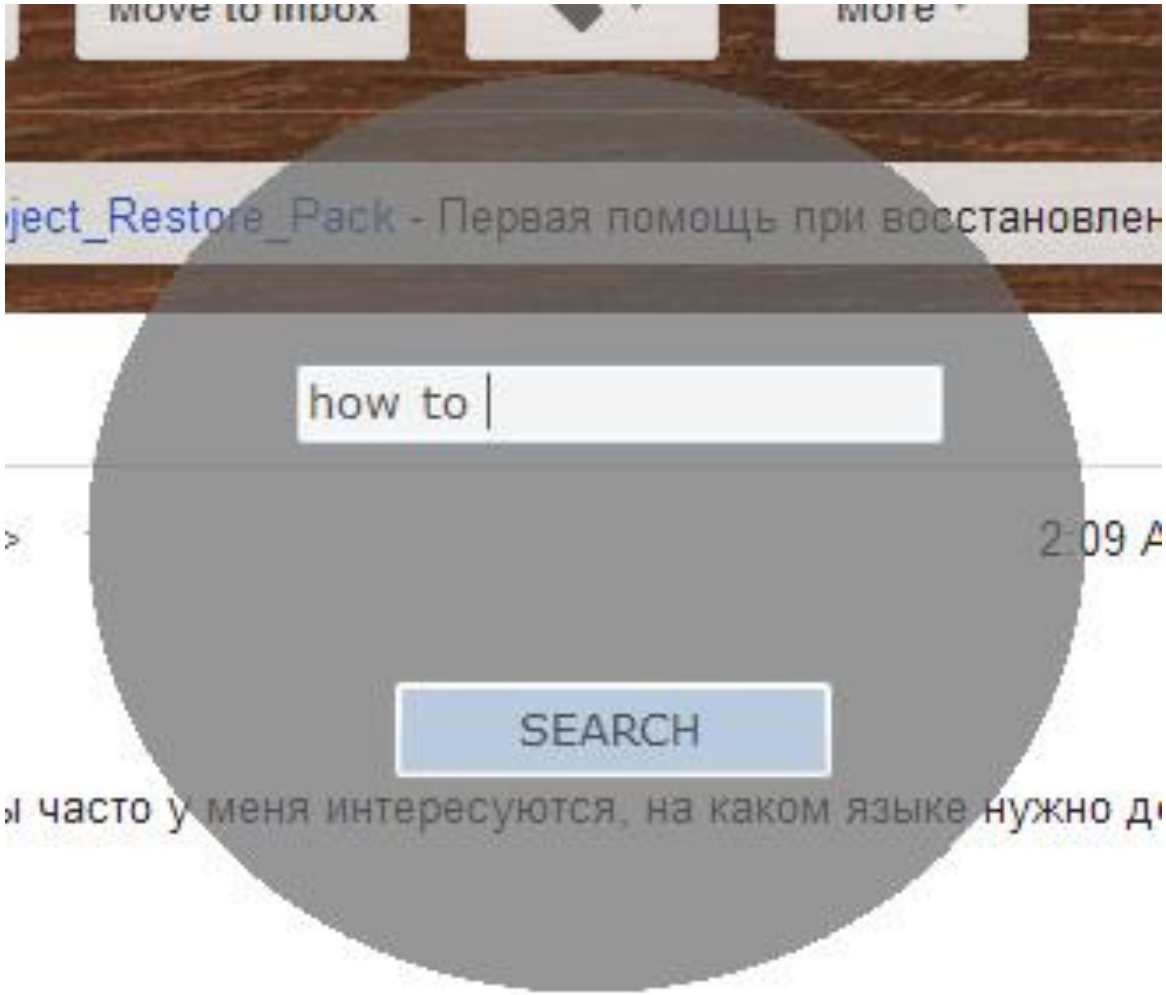
- At times you might want the startup form to be invisible at run time. For that, you can set Form's ***Visible*** property to false

Creating Nonrectangular WFs

- For advanced visual effects, you might want to create forms that are nonrectangular.
  - For example, you might want to create an oval form or a form in the shape of your company's logo.
- For that, set the ***Region*** property of the form in the ***Form_Load*** event handler.
  - The easiest way to create a nonrectagular region is to create a new instance of the ***GraphicsPath*** class and then create the new ***Region*** from it.

Do not forget to set ***FormBorderStyle*** *to* ***None.***

```
GraphicsPath myPath = new GraphicsPath();
myPath.AddEllipse(0, 0, this.Width, this.Height);
Region myRegion = new Region(myPath);
this.Region = myRegion;
```


Move to Inbox
More
ject_Restore_Pack - Первая помощь при восстановлен
how to
2:09 A
SEARCH
часто у меня интересуются, на каком языке нужно д

Container Controls

- ***Container controls*** are specialized controls that serve as a customizable container for other controls.
  - e. g. ***Panel, FlowLayoutPanel***, and ***SplitContainer***

- ***Container controls*** give the form logical and physical subdivisions that can group other controls into consistent UI subunits.
  - e.g. , you might contain a set of related ***RadioButton*** controls in a ***GroupBox*** control.

- Remember, that when a container control holds other controls, changes to the properties of the host control can affect the child controls
  - e.g. ***Enabled*** property

The Controls Collection

- Each form and container control has a property called ***Controls***, which represents the collection of controls contained by that form or control.

```
foreach (Control b in this.Controls)
{
    if(b is Button) b.Click += new EventHandler(b_Click);
    if (b is RadioButton) b.MouseClick += new MouseEventHandler(rb_MouseClick);
}
```

- Handlers:

```
void b_Click(object sender, EventArgs e)
{
    if(sender is Button)
        MessageBox.Show("haha, I am a Button, my name is  "+ ((Button)sender).Name);
}
void rb_MouseClick(object sender, MouseEventArgs e)
{
    if(sender is RadioButton)
        MessageBox.Show("haha, I am a Radio Button, my name is  " + ((RadioButton)sender).Name);
```

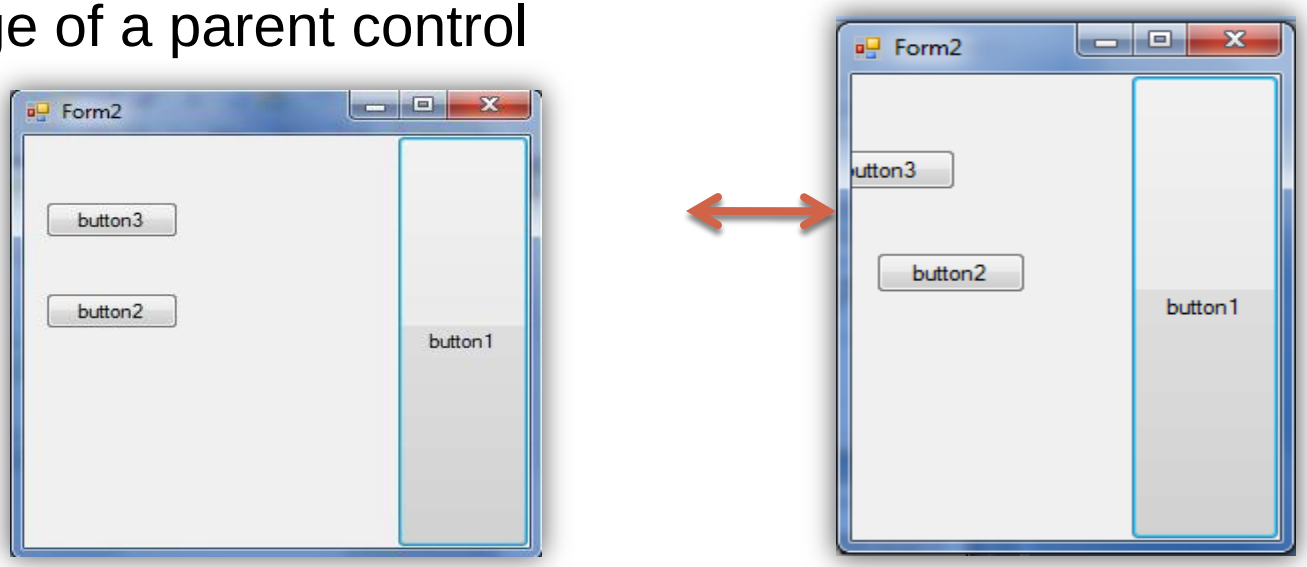
Adding controls at runtime

- To add a control to a form or container control at run time, manually instantiate a new control and add it to the ***Controls*** collection of the form

```
Button aButton = new Button();
aButton.Location = new Point(20, 20);
aButton.Text = "Test Button";
Panel1.Controls.Add(aButton);
this.Controls.Add(aButton);
```

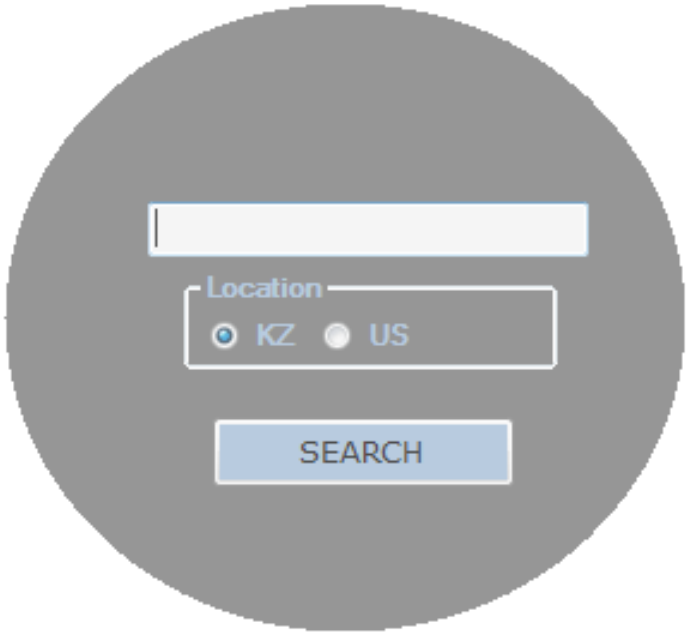
Anchor, Dock

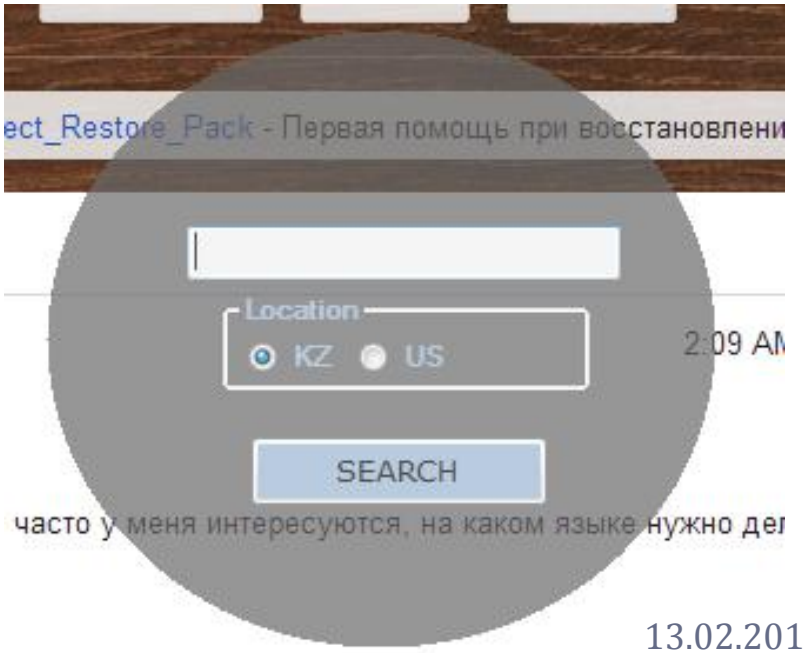
- The ***Anchor*** and ***Dock*** properties of a control dictate how it behaves inside its form or parent control.
- The ***Anchor*** property allows you to define a constant distance between edges of a control and edges of a form or other container.
  - Thus, if a user resizes a form at run time, the control edges will always maintain a specific distance from the edges.
- The ***Dock*** property enables you to attach your control to the edge of a parent control



GroupBox

- The ***GroupBox*** control is a container control that appears as a subdivision of the form surrounded by a border.
  - It also provide a caption (***Text*** property)
  - The most common use for ***GroupBox*** controls is for grouping ***RadioButton*** controls (within GB they are *mutually exclusive*)





Panel

- The ***Panel*** control creates a subsection of a form that can host other controls.
  - Scrollable control

PROPERTY	DESCRIPTION
AutoScroll	Determines if the *Panel* will display scroll bars when controls are hosted outside the visible bounds of the *Panel*. Scroll bars are displayed when this property is set to *True* and are not displayed when it is set to *False*.
BorderStyle	Represents the visual appearance of the *Panel* border. This property can be set to *None*, which indicates no border; *FixedSingle*, which creates a single-line border; or *Fixed3D*, which creates a border with a three-dimensional appearance.

FlowLayoutPanel

- ***FlowLayoutPanel*** dynamically repositions the controls it hosts when it is resized at design or run time.
  - This provides a great aid to UI design because control positions are automatically adjusted as the size and dimensions of the ***FlowLayoutPanel*** are adjusted
  - Resembles dynamic realignment of the UI much like an HTML page
- Default direction – ***left - > right***
- You can set flow break :

```
Flp.SetFlowBreak(aButton, true);
```

TableLayoutPanel

- ***TableLayoutPanel*** is essentially a table that provides cells for the individual hosting of controls.
- At run time the ***CellBorderStyle*** property determines the appearance of the cells

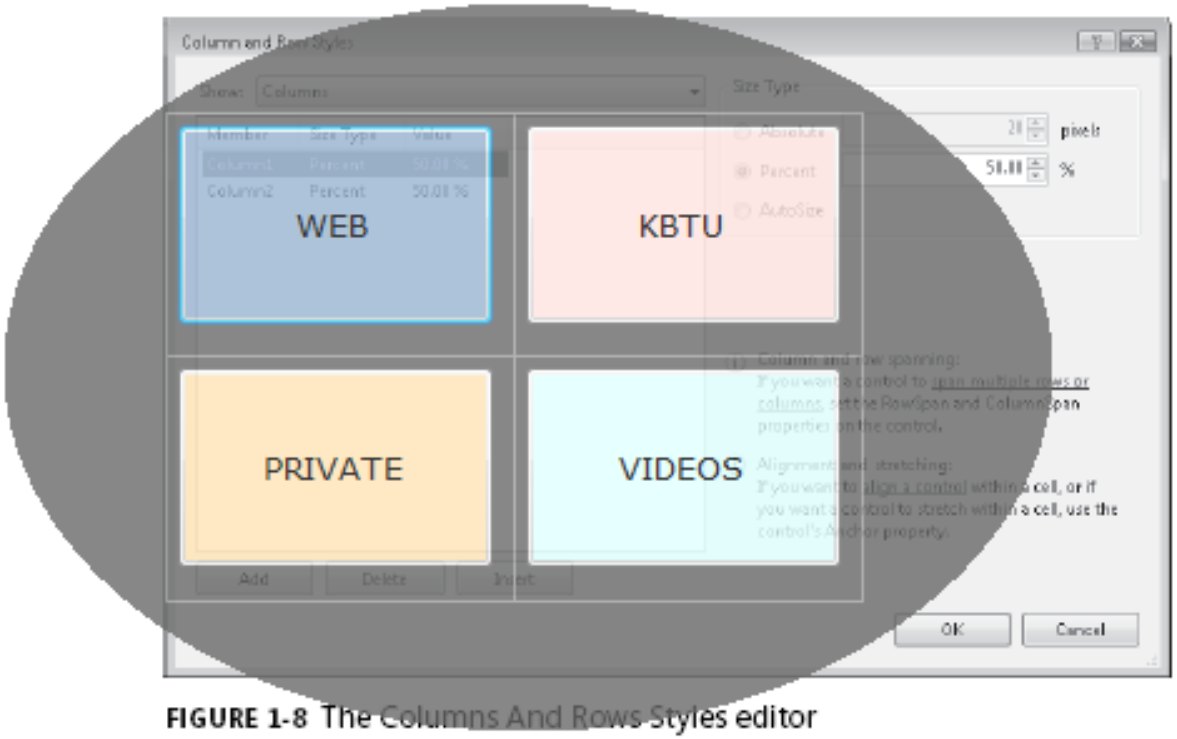
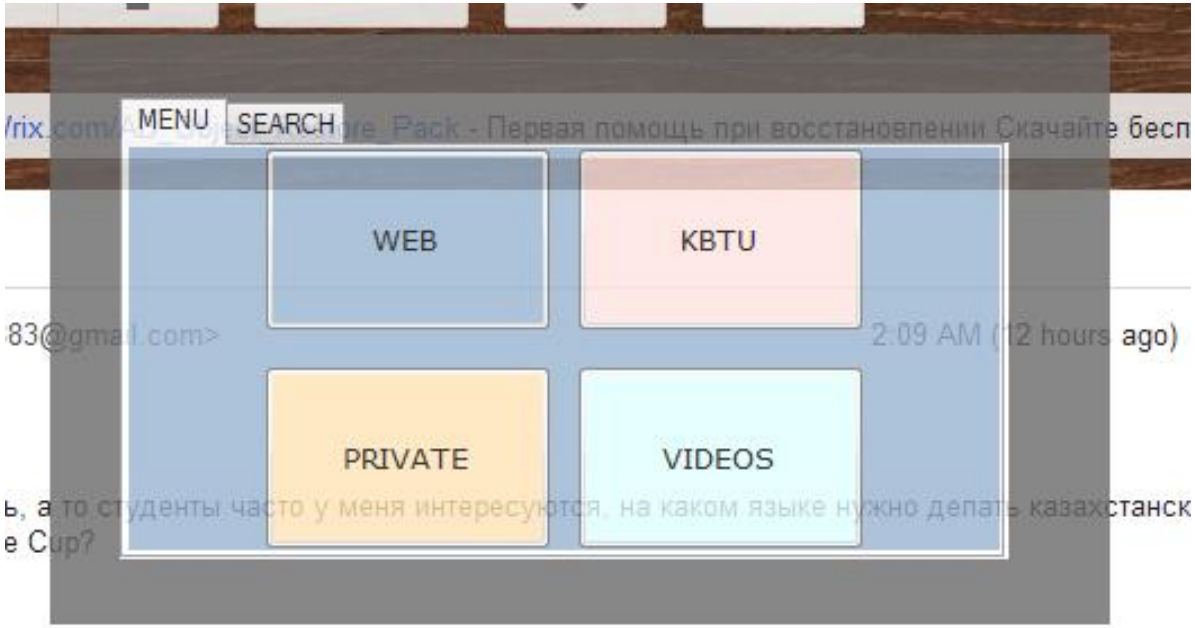


FIGURE 1-8 The Columns And Rows Styles editor

You can alter column and row size styles with this editor. Column an

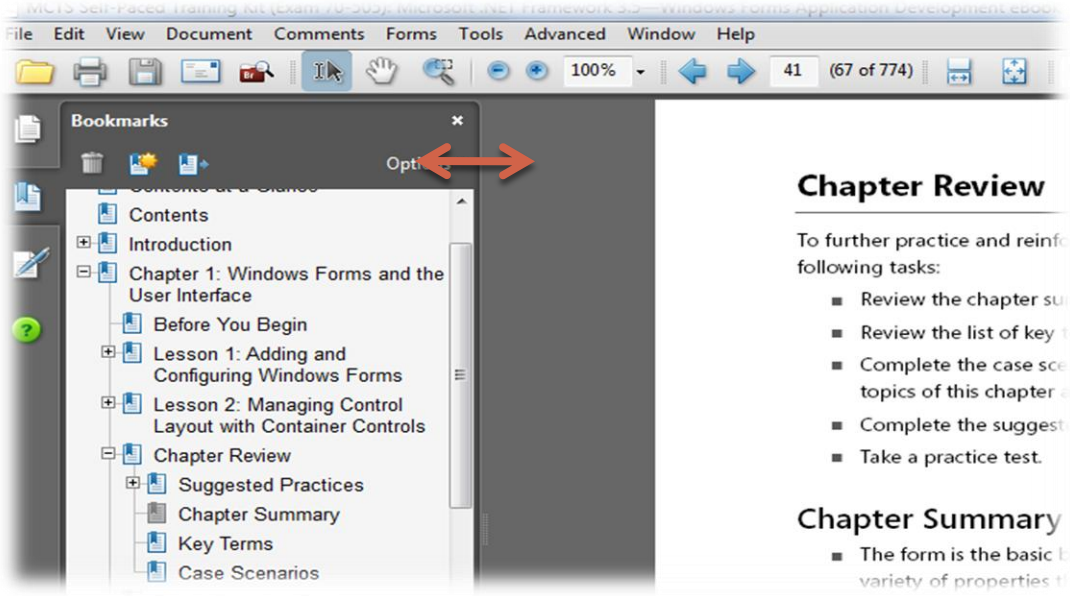
TabControl

- Enables you to group sets of controls in tabs
  - For example, you might create property pages for an application in which each page represents the properties of a specific component.
- The ***TabControl*** serves as a host for one or more ***TabPage*** controls, which themselves contain controls.



SplitContainer

- Creates a subsection of the form where a **Splitter** divides the ***SplitContainer*** into two **SplitterPanel** controls that function similarly to **Panel** controls.
  - They are accessible via Panel1 and Panel2 properties
- The user can grab the **Splitter** with the mouse and move its location, thus changing the relative size of each **SplitterPanel**
- **Properties:** ***Orientation, IsSplitterFixed. Panel1Collapsed***



Summary

- The ***form*** is the basic building block of Windows Forms applications. Forms provide a variety of properties that you can use to affect the appearance of the user interface, including ***Text, BorderStyle, Size, Opacity***, and the behavior of the UI, such as ***Windowstate*** and ***TopMost.***
- Forms are generally rectangular, but you can create nonrectangular forms by setting the ***Region*** property to a nonrectangular region.
- Container controls can host and help manage layout of individual controls.
- The ***SplitContainer*** control can be used to create dynamically sizable sections of a form, each of which contains its own controls.
- Controls can be added to a form at design time by selecting a control from the Toolbox or they can be added dynamically at run time.

Configuring Controls

- All controls inherit from the base class ***Control*** and, as such, share a variety of properties relating to size, location, and other general aspects of controls.

- The ***Layout toolbar*** provides a quick and easy way to perform many of the control layout tasks required at design time.
- When you are creating forms that contain several container controls, the ***Document Outline*** window can be useful for allocating controls between the various containers.
  - The ***Document Outline*** window graphically displays all of the controls and container controls that reside in a form.
  - With the mouse, you can grab controls in the ***Document Outline*** window and move them from one container to another.
  - You can also delete controls there
  - **View -> Other Windows - > Document Outline**

Best Practices for User Interface Design

- How your UI is composed influences how easily users can learn and use your application.
- Primary considerations for UI design include:
  - ***Simplicity***
    - expose only the functionality needed at each stage of the application.
  - ***Position of controls***
    - The location of controls on your UI should reflect their relative importance and frequency of use
  - ***Consistency***
    - UI should exhibit a consistent design across each form in your application. Consistency is created from the use of colors, fonts, size, and types of control.
  - ***Aesthetics***
    - Whenever possible, a UI should be inviting and pleasant

Types of controls

- **Text display controls**
  - ***Label*** and ***LinkLabel*** are used to convey read-only information to the user
  - e.g. , labels are frequently used to display an informative string beside a control, such as “First Name” beside a ***TextBox***
    - You can use ***Label*** controls to define access keys for other controls. Access keys are keys that, when pressed in combination with the Alt key, move the focus to the desired control.
  - ***LinkLabel*** control allows you to create a Web-style link in your form that opens a Web page or performs some other action when clicked.
- **Command controls**
  - ***Button*** control are used to execute tasks
  - ***Button*** can also serve as an accept or cancel button
- **Text edit controls**
  - ***TextBox*** controls are used both to display text to the user and to receive textual input.
  - The ***MaskedTextBox*** control allows you to display text in a preset format and validate user input against a format.

Text Box

- ***TextBox*** allows you to receive text from and display text to the user.
  - You can create text boxes that can display multiline text
  - You can create text boxes that display a password character instead of the actual text.

PROPERTY	DESCRIPTION
AutoCompleteCustom-Source	Holds a string collection that contains autocomplete data when the *AutoCompleteMode* is set to a value other than *None* and the *AutoCompleteSource* is set to *Custom*.
AutoCompleteMode	Sets the *AutoComplete* mode of the control. Possible values are *None, Append, Suggest,* or *SuggestAppend*.
AutoCompleteSource	Sets the source for autocomplete data. Can be set to any of a variety of system sources or to a custom source provided by the *AutoCompleteCustomSource* property.
CharacterCasing	Indicates the casing of the characters in the *TextBox* control. Possible values are *Normal, Upper,* or *Lower*.
MultiLine	Indicates whether the *TextBox* can contain only a single line of text or multiple lines.
PasswordChar	Sets the password character to be displayed in the *Textbox* instead of the actual text.
ReadOnly	Indicates whether the text in the *TextBox* can be edited.
ScrollBars	Indicates whether scroll bars are displayed in the *TextBox* when the *MultiLine* property is set to *True*.
Text	Gets or sets the text contained in the *TextBox*.

Masked Text Box

- The ***MaskedTextBox*** control is a modified ***TextBox*** that allows you to define a preset pattern for accepting or rejecting user input.
- The ***Mask*** property allows you to specify required or optional characters or specify whether input characters are letters or numbers and apply formatting for the display of strings.

MASK STRING	INPUT TEXT	DISPLAYED TEXT
(999)-000-0000	1234567890	(123)-456-7890
00/00/0000	07141969	07/14/1969 – Note that the actual date separator displayed is determined by the control's FormatProvider.
$99,999.00	1234567	$12,345.67 – Note that the actual currency symbol, thousands separator, and decimal separator is determined by the control's FormatProvider.
LL>L|LLL<LL	abcdABCD	abCdABcd

Summary

- Controls are visual components that provide functionality designed to enable common tasks.
- The ***Button*** control is designed to accept user commands and execute code when clicked.
- ***Label*** controls are primarily used to display read-only text and can be used to create access keys for other controls.
- The ***LinkLabel*** control allows you to create Web-style links that open Web pages or other forms when clicked.
- The ***TextBox*** control is used to receive user input as well as to display text. ***TextBox*** controls can be either single-line or multiline.
- The ***MaskedTextBox*** enables you to specify a format for text display or user input. It enables you to configure how that format restricts user input