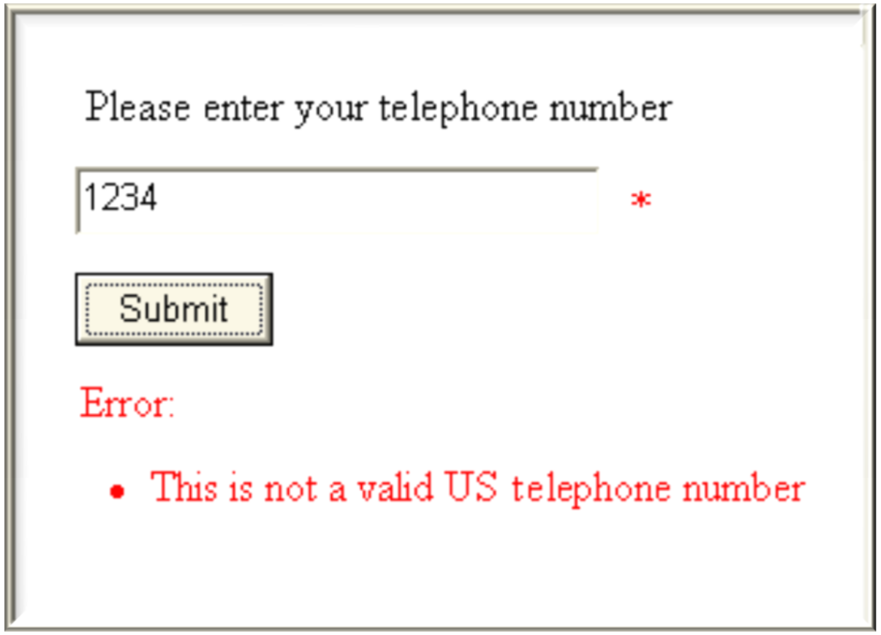
Validation & Rich Controls

What Is Input Validation?

- Verifies that a control value is correctly entered by the user
- Blocks the processing of a page until all controls are valid
- Avoids spoofing or the addition of malicious code


Please enter your telephone number

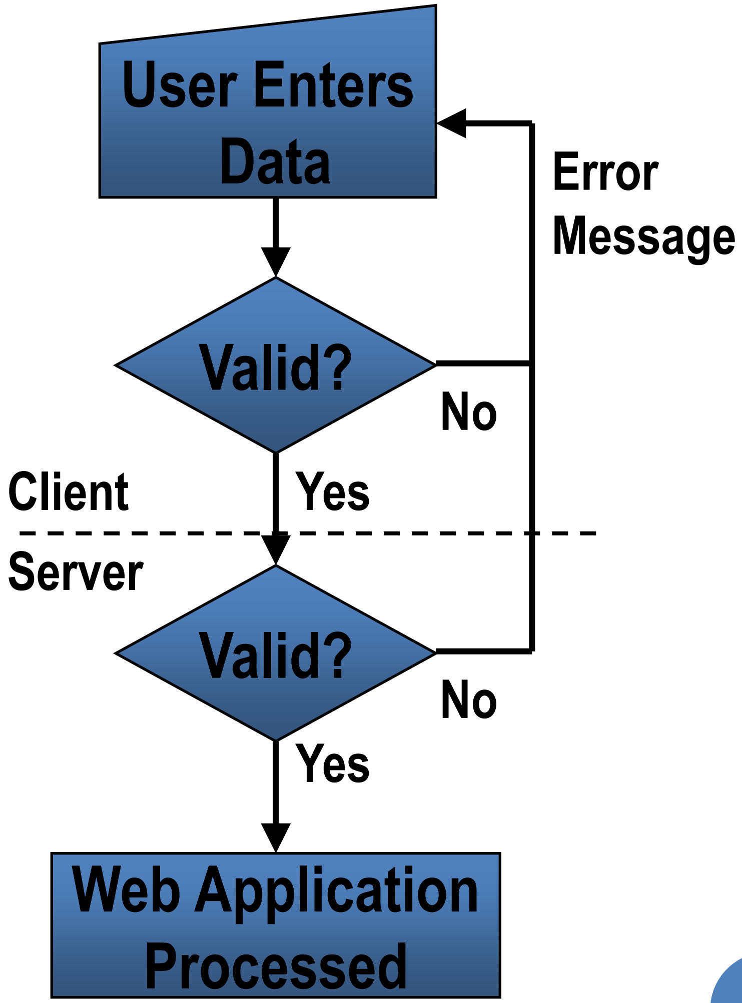
1234 *

Error:

- This is not a valid US telephone number

Client-Side and Server-Side Validation

- **ASP.NET can create both client-side and server-side validation**
- **Client-side validation**
  - Dependent on browser version
  - Instant feedback
  - Reduces postback cycles
- **Server-side validation**
  - Repeats all client-side validation
  - Can validate against stored data



ASP.NET Validation Controls

- ASP.NET provides validation controls to:
- Compare values
- Compare to a custom formula
- Compare to a range

ASP.NET Validation Controls

- ASP.NET provides validation controls to:
- Compare to a regular expression pattern
- Require user input
- Summarize the validation controls on a page

Validation Controls

- All validation control classes derive from *BaseValidator*.
- Each validation control can be bound to a single input control.
- More than one validation control to the single input control are allowed.
- *Button* control has *CausesValidation* property. If set to *false*, button click do not fire any validation.

BaseValidator Properties

- ControlToValidate
  - Identifies the control that is to validated.
- ErrorMessage
  - If validation fails, the validator control can display a error message.
- Display
  - Allows you to configure whether this error message will be added dynamically as needed (Dynamic) or whether an appropriate space will be reserved for the message (Static).

BaseValidator Properties

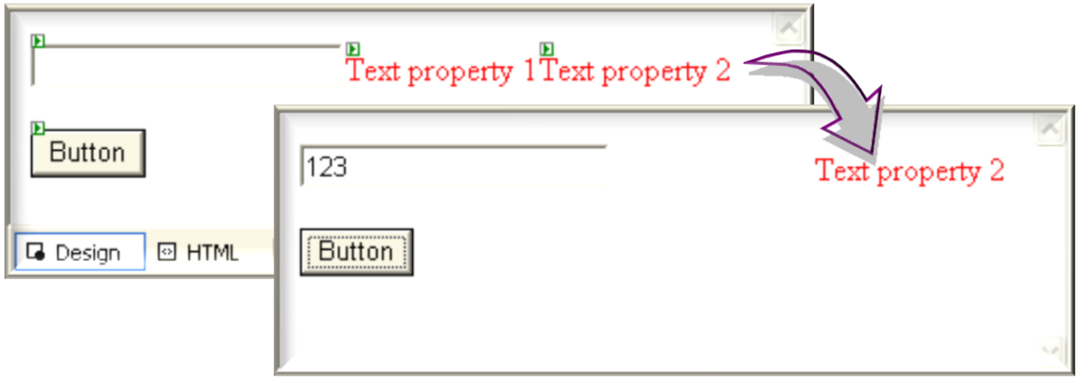
- IsValid
  - Returns *true* or *false* depending on the valiadtion success or failure.
- EnableClientSideScript
  - If set to true, ASP.NET will add JavaScript and DHTML code to allow client-side validation.
- ValidationGroup
  - Gets or sets the validation group that this control belongs to.

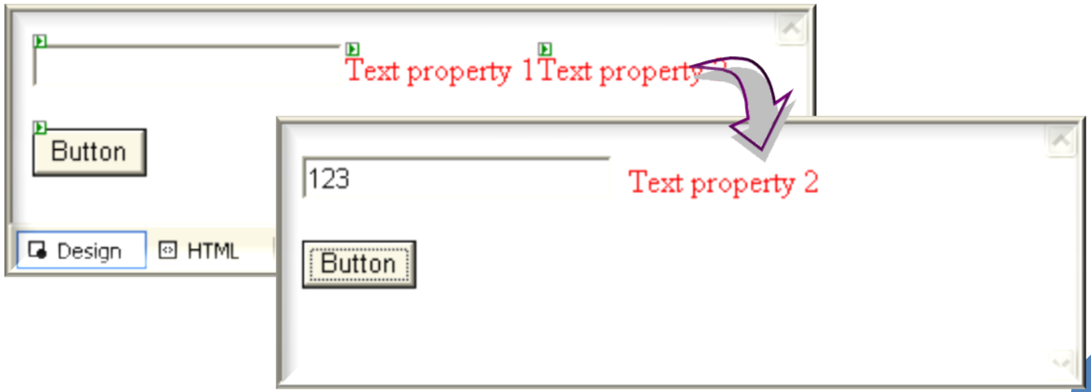
BaseValidator Properties

- SetFocusOnError
  - Indicates whether the focus is moved to the control where validation failed.
- Text
  - Gets or sets the description displayed for the validator in lieu of the error message.
  - Do not replace the contents of *ErrorMessage* in the summary text.

POSITIONING VALIDATION CONTROLS ON A WEB FORM

- Create error messages
- Select display mode
  - Static
  - Dynamic





INPUT VALIDATION CONTROLS

- RequiredFieldValidator
  - **InitialValue →** Specifies the initial value of the input control. Default value is empty string.

- CompareValidator
  - **ValueToCompare→**Indicates the value to compare the user's input against.
  - **Type →**Specifies the data type of value.
  - **ControlToCompare →**Represents the ID of the control to compare with the current user's entry.
  - **Operator →** Specifies the comparison operation to perform. Default is *equal*.

Input Validation Controls

- RangeValidator
  - MinimumValue
  - MaximumValue
  - Type
- RegularExpressionValidator
  - Used when input must conform to a pre-defined pattern
  - **ValidationExpression** → Set with the regular expression, which will be used to validate the input.

CustomValidator Control

- Lets user to define the validator with custom validation logic.

- To set up the Custom validator function at client side, use *ClientValidationFunction*.

- One can program for *ServerValidate* event with server-side validation logic.

CustomValidator Client Side Function

- The client validation function takes a mandatory signature.

function <funcname>(source, arguments)

{ … }

- source → References the validator control.

- Arguments → References an object with two properties, *IsValid* and *Value*.

ServerValidate Event Handler

- The event handler has *ServerValidateEventArgs* argument.

- It has two properties

  - IsValid →Set to false, if validation fails.

  - Value →Value entered in the input control.

ValidationSummary Control

- *ValidationSummary* control is a label that summarizes and displays all the validation error messages found on a Web page.
- *DisplayMode* property sets the output format.
- *ShowSummary* property, if true, displays messages in page itself.
- *ShowMessageBox* property, if true, displays messages in message box.

Page.IsValid Property

- True if all validations are successful.

```
private void cmdSubmit_Click(object s, System.EventArgs e)
{		if (Page.IsValid)
		{		Message.Text = "Page is Valid!";
				// Perform database updates or other logic here
		}
}
```

```
Sub cmdSubmit_Click(s As Object, e As EventArgs)
		If Page.IsValid Then
				Message.Text = "Page is valid!"
				' Perform database updates or other logic here
		End If
End Sub
```

Validation Groups

- If one has set of input and validation controls and two buttons on the form, clicking either button will always validate all controls.

- Define it for all the validation controls that you want to group together, and then assign the same name to the *ValidationGroup* property of the button.

Rich Controls

- Rich controls are web controls that model complex user interface elements.

- Rich Controls

  - DropDownList
  - CheckBoxList
  - AdRotator
  - Calendar
  - MultiView, View
  - Wizard

AdRotator Control

- A banner ad that displays one of a set of images based on a predefined schedule that's saved in an XML file.
- The AdRotator requires
  - Advertisement File
    - Stores the list of image files in a special XML file.

Advertisement File

```
<?xml version="1.0" encoding="utf-8" ?>
<Advertisements>
  <Ad>
    <ImageUrl>./Images/logo_iconnect.jpg</ImageUrl>
    <NavigateUrl>http://www.iconnectgroup.com/</NavigateUrl>
    <AlternateText>iConnect Software India</AlternateText>
    <Impressions>1</Impressions>
    <Keyword>Software</Keyword>
  </Ad>
  <Ad>
            …
  </Ad>
</Advertisements>
```

A keyword that identifies a group of advertisements.

AdRotator Properties & Event

- AdvertisementFile
- Target → _top, _self, _parent, _blank
- KeywordFilter →Keyword for limiting selection of ads.
- DataSourceID → Specifies a data source control ID to use. Do not set the ads file.
- *AdCreated* Event
  - One parameter is of type *AdCreatedEventArgs*.
  - *AdCreatedEventArgs* contains information about the image, navigation URL, alternate text, and any custom properties.
  - One can control the Ad to show.

Calendar Control

- Displays one-month calendar.
- Allows navigation from month-to-month.
- One can fully customize the control.
- *SelectionMode* property decides what a user can select --a single date, week, or month.
- *SelectedDate* property is used to get or set date selection.

Calendar Events

- DayRender
  - Fired when each date cell is created.
  - Used to customize particular day cell output.
- SelectionChanged
  - Fired when selected date changed.
- VisibleMonthChanged
  - Fired when one navigates to new month.

Demo: Using Calendar

- Customizing Calendar
- Events Calendar
- Restricting Selection

MultiView and View Controls

- On occasion, one wants an area of a page to display material that may vary depending on circumstances at the moment.
  - An ***area of the page*** may show different info depending on whether the ***user*** is an ***administrator***, ***developer***, or ***guest user***.
  - A ***tabbed control*** may show different information depending on which ***tab*** has been clicked.
- A ***MultiView*** control is useful in such situations.

MultiView & View Controls

- The ***View*** control represents one ***View*** than can be displayed in a ***MultiView*** control.
- The ***MultiView control*** is merely a ***container*** for several ***View controls only one*** of which is ***visible at a time***.
- ***MultiView*** takes care of coordinating all the views with the help of ***ActiveViewIndex*** property or ***SetActiveView*** method.

MultiView & View Controls

```
<asp:MultiView runat="server" id="Tables">
        <asp:View runat="server" id="Employees">
        ...
        </asp:View>
        <asp:View runat="server" id="Products">
        ...
        </asp:View>
        <asp:View runat="server" id="Customers">
        ...
        </asp:View>
 </asp:MultiView>
```

Postbacks when the user clicks buttons or links embedded in the current view.

Wizard Control

- Wizards represent a single task, and the user moves linearly through them, moving from the current step to the one immediately following it.
- *Wizard* control supports nonlinear navigation.
- Wizard control supplies navigation buttons and a sidebar with links for each step on the left.

Wizard Steps

- To create wizard, simply define the steps and their content using <asp:WizardStep> tags.

- Title→ Descriptive name of the step. Acts as a link text in sidebar.

Wizard Steps

- AllowReturn → Specifies whether the user can return to this step or not.

- StepType →Type of a wizard step.

  - Start → displays Next button only
  - Step → displays perv & next button
  - Finish → displays prev & Finish button.
  - Complete → No sidebar, no buttons.
  - Auto →Position in collection determines which buttons to show.

Wizard Properties

- DisplaySideBar
  - Specifies visibilty of side bar.
- ActiveStepIndex
  - Current visible step index.
- DisplayCancelButton
  - Determines whether to display cancel button or not.
- CancelDestinationPageUrl
  - If Cancel button clicked, take to URL specified.

Wizard Events

- ActiveStepChanged
  - Occurs when the control switches to a new step.
- CancelButtonClick
  - Occurs when the Cancel button is clicked.
- FinishButtonClick
  - Occurs when the Finish button is clicked.
- NextButtonClick/PreviousButtonClick
  - Occurs when the Next or Previous button is clicked on any step.
- SideBarButtonClick
  - Occurs when a button in the sidebar area is clicked.