

# **Validation**

Asst. Prof. Dr. Özgü Can

# Validation

- Data usually flows in two directions:
  1. It either comes from the server and is sent to the end user's browser.
    - Files and databases

or

2. The data is entered by the user and sent to the server to be processed or stored.
  - Page requests, contact forms, shopping cart scenarios

# Validation

- To prevent your system from receiving *invalid* data, it's important to **validate this data** before you allow your system to work with it.
- Gathering data from the user:
  - **GET**: Data is appended to the actual address of the page being requested.
  - **POST**: Data is sent in the body of the request for the page.

# GET

- Data is added to the requested address for a page.
- You can retrieve it using the **QueryString** property of the **Request** object.

<http://www.MyWebSite.com/Reviews/ViewDetails.aspx?ReviewId=34&CategoryId=3>

- To access individual items in the query string, you can use the **Get** method of the **QueryString** collection:

```
// Assigns the value 34 to the reviewId variable
int reviewId = Convert.ToInt32(Request.QueryString.Get("ReviewId"));
// Assigns the value 3 to the categoryId variable
int categoryId = Convert.ToInt32(Request.QueryString.Get("CategoryId"));
```

# POST

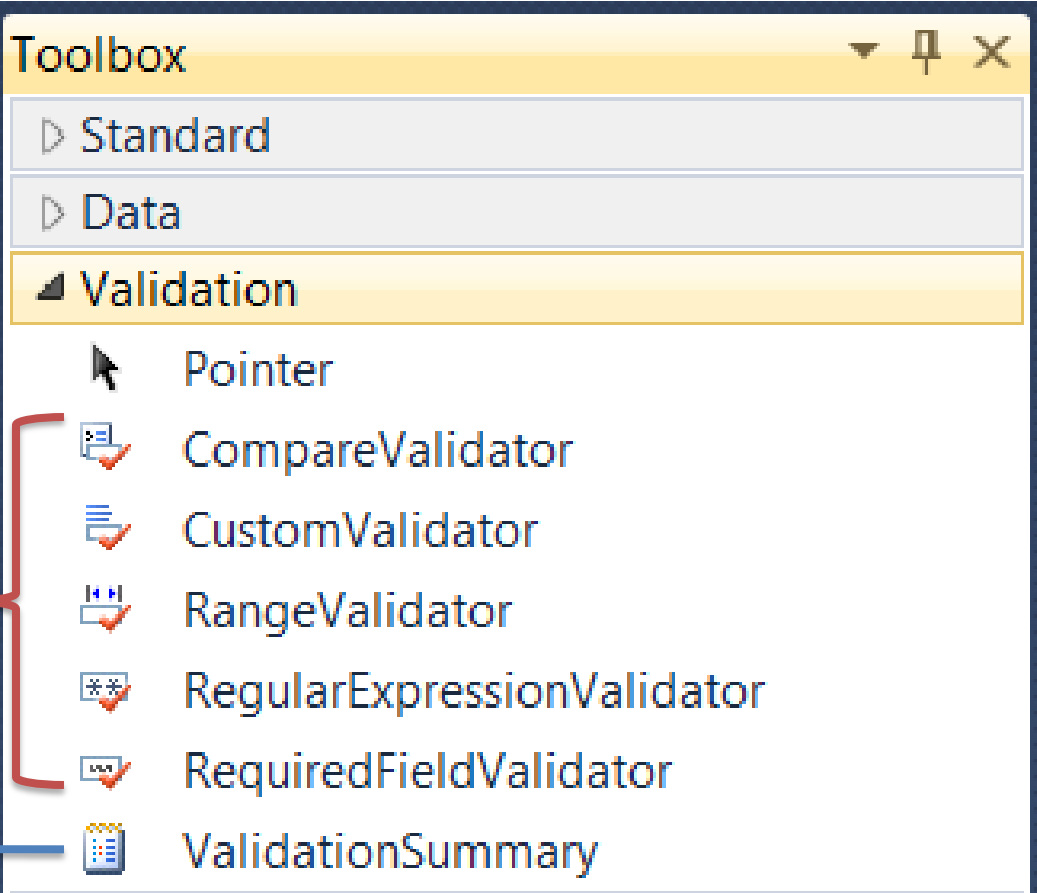
- Gets its data from a form with controls that have been submitted to the server.
- A **TextBox** called **Age** to hold the user's age and a **Button** to submit that age to the server.
- In the **Button** control's **Click** event:

```
int age = Convert.ToInt32(Age.Text);           // age now holds the user's age
```

- What happens when a user submits invalid data, either deliberately or by accident?
  - **Code will crash & throw an exception**
  - **Validate** all the data that is being sent to server.

# ASP.NET Validation Controls

*Never trust user input!*



The screenshot shows the ASP.NET Toolbox with the 'Validation' category expanded. The controls listed are: Pointer, CompareValidator, CustomValidator, RangeValidator, RegularExpressionValidator, RequiredFieldValidator, and ValidationSummary. A red bracket groups the first six validators, and a blue arrow points to the ValidationSummary control.

**Perform the actual validation**

**Provides feedback to the user about any errors made in the page**

# Validation Controls

- Check the input at the **client** and at the **server**.
- ***Never*** rely on client-side validation as the only solution to validation.
- Server-side validation is the ***only*** real means of validation.

PROPERTY	DESCRIPTION
Display	This property determines whether or not the hidden error message takes up space. With the <code>Display</code> set to <code>Static</code> , the error message takes up screen estate, even when it is hidden. This is similar to the CSS setting <code>visibility: hidden</code> you saw in earlier chapters. The <code>Dynamic</code> setting hides the error message using <code>display: none</code> until it needs to be displayed. With a setting of <code>None</code> , the error message is not visible at all. This is useful if you are using a <code>ValidationSummary</code> .
CssClass	This property enables you to set the CSS <code>class</code> attribute that is applied to the error message text.
ErrorMessage	This property holds the error message used in the <code>ValidationSummary</code> control. When the <code>Text</code> property is empty, the <code>ErrorMessage</code> value is also used as the text that appears on the page.
Text	The <code>Text</code> property is used as the text that the validation control displays on the page. This could be an asterisk (*) to indicate an error, or text like "Enter your name."
ControlToValidate	This property contains the server ID of the control that needs to be validated.
EnableClientScript	This property determines whether the control provides validation at the client. The default is <code>True</code> .
SetFocusOnError	This property determines whether client-side script gives the focus to the first control that generated an error. This setting is <code>False</code> by default.
ValidationGroup	Validation controls can be grouped together, enabling you to perform validation against a selection of controls. All controls with the same <code>ValidationGroup</code> are checked at the same time, which means that controls that are not part of that group are not checked. Consider, for example, a login page with a Login button and fields for a user name and password. The same page may also contain a search box that enables you to search the site. With the <code>ValidationGroup</code> , you can have the Login button validate the user name and password boxes, whereas the Search button triggers validation for just the search box.
IsValid	You don't typically set this property at design time, but at runtime it provides information about whether the validation test has passed.



# RangeValidator

- Enables you to check whether a value falls within a certain range.

PROPERTY	DESCRIPTION
MinimumValue	This property determines the lowest acceptable value. For example, when checking an integer number between 1 and 10, you set this property to 1.
MaximumValue	This property determines the highest acceptable value. For example, when checking an integer number between 1 and 10, you set this property to 10.
Type	This property determines the data type that the validation control checks. This value can be set to <code>String</code> , <code>Integer</code> , <code>Double</code> , <code>Date</code> , or <code>Currency</code> to check the respective data types.

```
<asp:RangeValidator ID="RangeValidator1" runat="server"  
    ControlToValidate="Rate" ErrorMessage="Enter a number between 1 and 10"  
    MaximumValue="10" MinimumValue="1" Type="Integer" />
```

# RegularExpressionValidator

- Enables you to check a value against a regular expression that you set in the **ValidationExpression** property of the control.

```
<asp:RegularExpressionValidator ID="RegularExpressionValidator1" runat="server"
    ControlToValidate="Email" ErrorMessage="Enter a valid e-mail address"
    ValidationExpression="\w+([-+.']\w+)*@\w+([-.\w+)*\.\w+([-.\w+)*" />
```

# CompareValidator

- Enables you to compare the value of one control to another value.
  - Sign-up forms where users have to enter a password twice to make sure they type the same password both times.
- You can also compare against a constant value.

# CompareValidator

PROPERTY	DESCRIPTION
ControlToCompare	This property contains the ID of the control that the validator compares against. When this property is set, ValueToCompare has no effect.
Operator	This property determines the type of compare operation. For example, when Operator is set to Equal both controls must contain the same value for the validator to be considered valid. Similarly, you have options like NotEqual, GreaterThan, and GreaterThanEqual to perform different validation operations.
Type	This property determines the data type that the validation control checks. This value can be set to String, Integer, Double, Date, or Currency to check the respective data types.
ValueToCompare	This property enables you to define a constant value to compare against. This is often used in agreements where you have to enter a word like Yes to indicate you agree to some condition. Simply set the ValueToCompare to the word Yes and the ControlToValidate to the control you want to validate and you're done. When this property is set, make sure that the ControlToCompare property is empty because that will otherwise take precedence.

```
<asp:CompareValidator ID="CompareValidator1" runat="server"
    ControlToCompare="ConfirmPassword" ControlToValidate="Password"
    ErrorMessage="Your passwords don't match" />
```

# CustomValidator

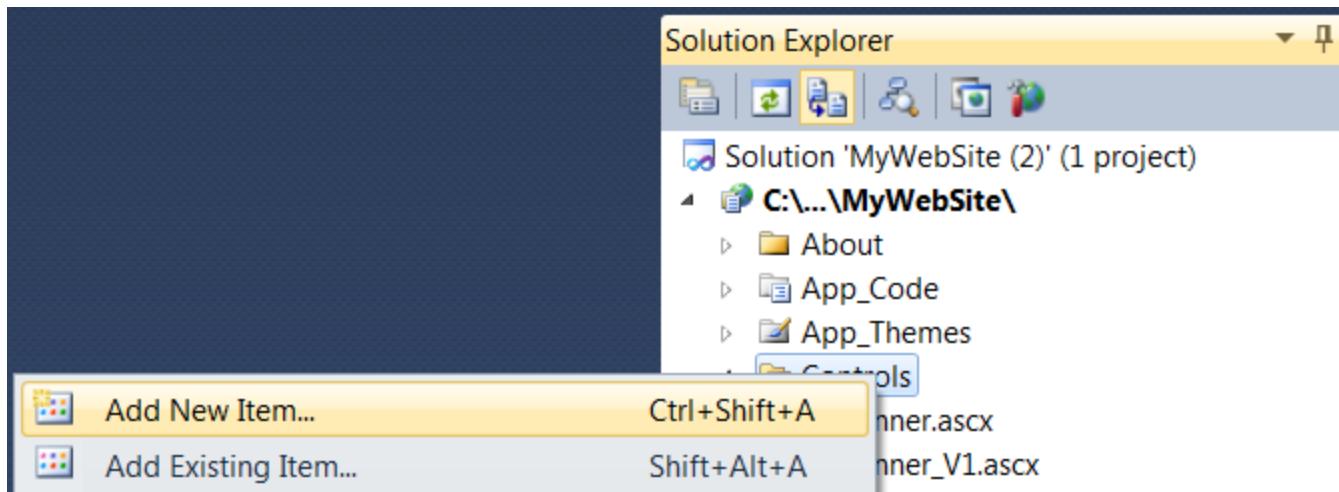
- Enables you to write custom validation functions for both the **client** (in JavaScript) and the **server** (using VB.NET or C#).

# ValidationSummary

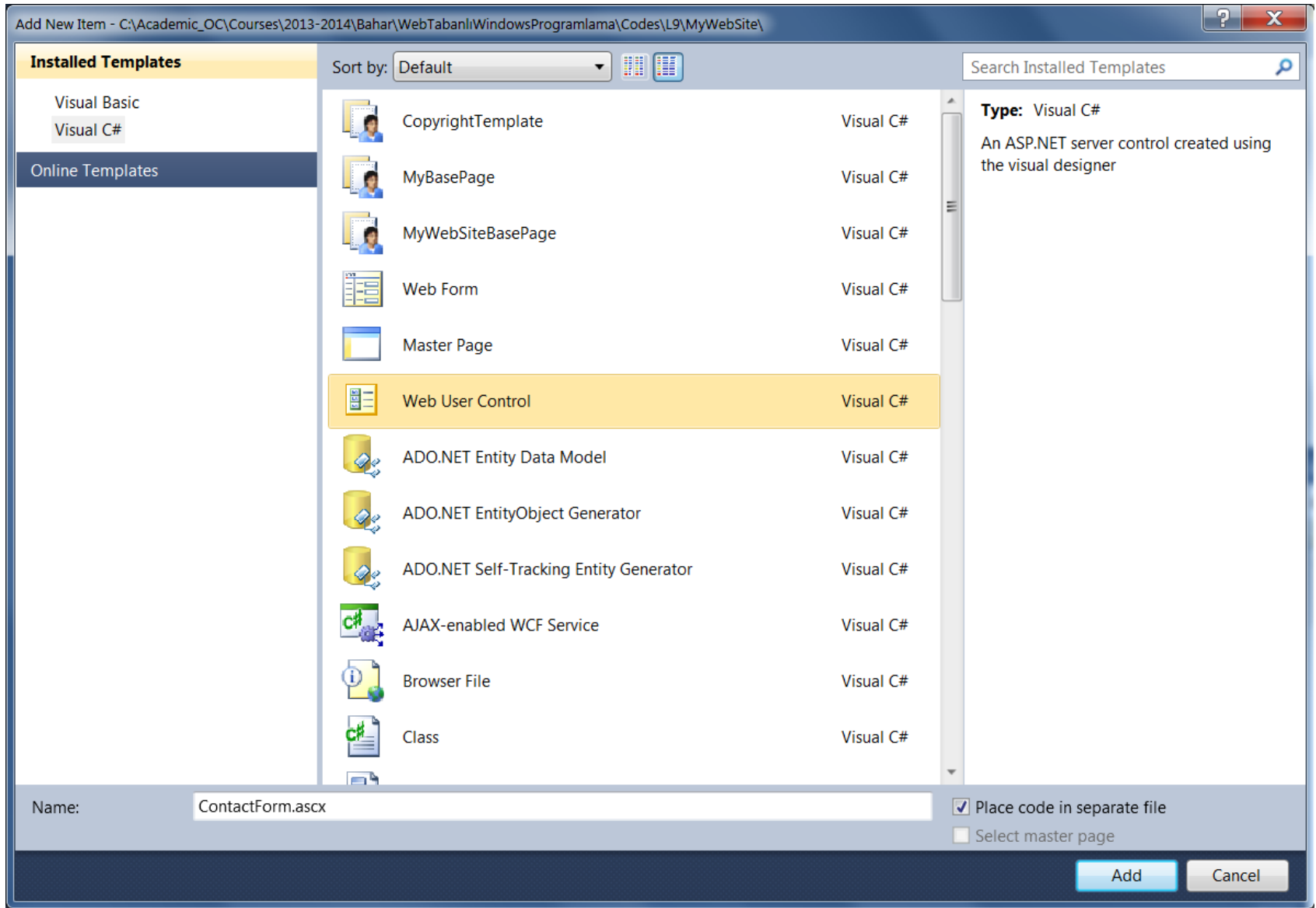
- Display errors.
- Provides the user with a list of errors that it retrieves from the individual validation control's **ErrorMessage** properties.

# Example

- In Controls folder → Add New Item



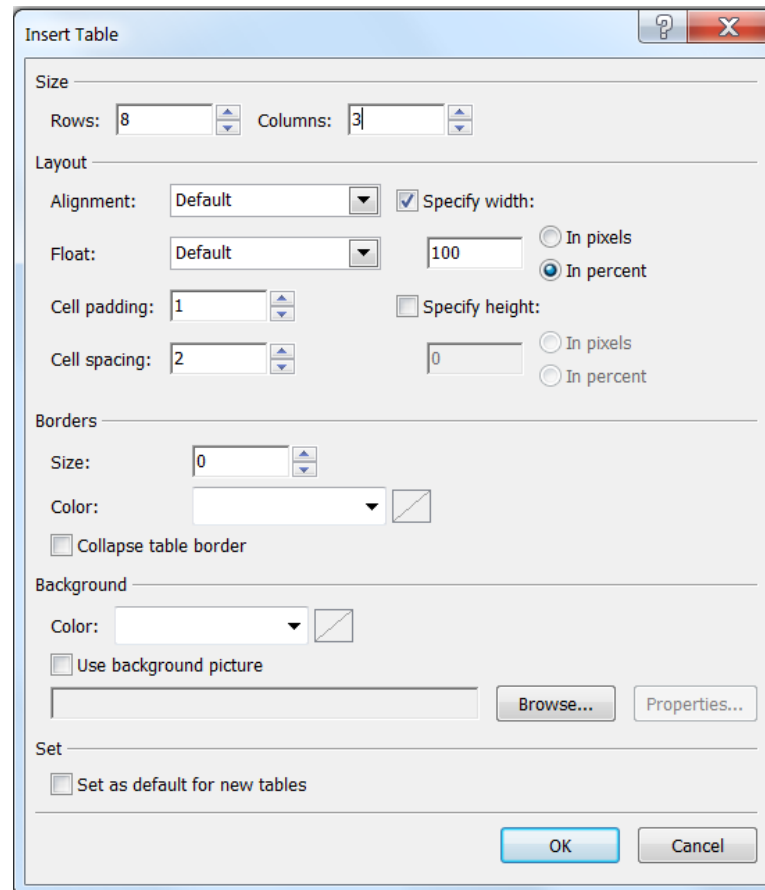
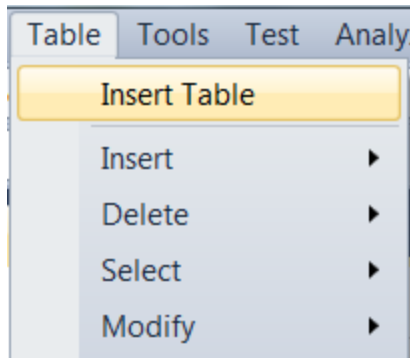
# Example





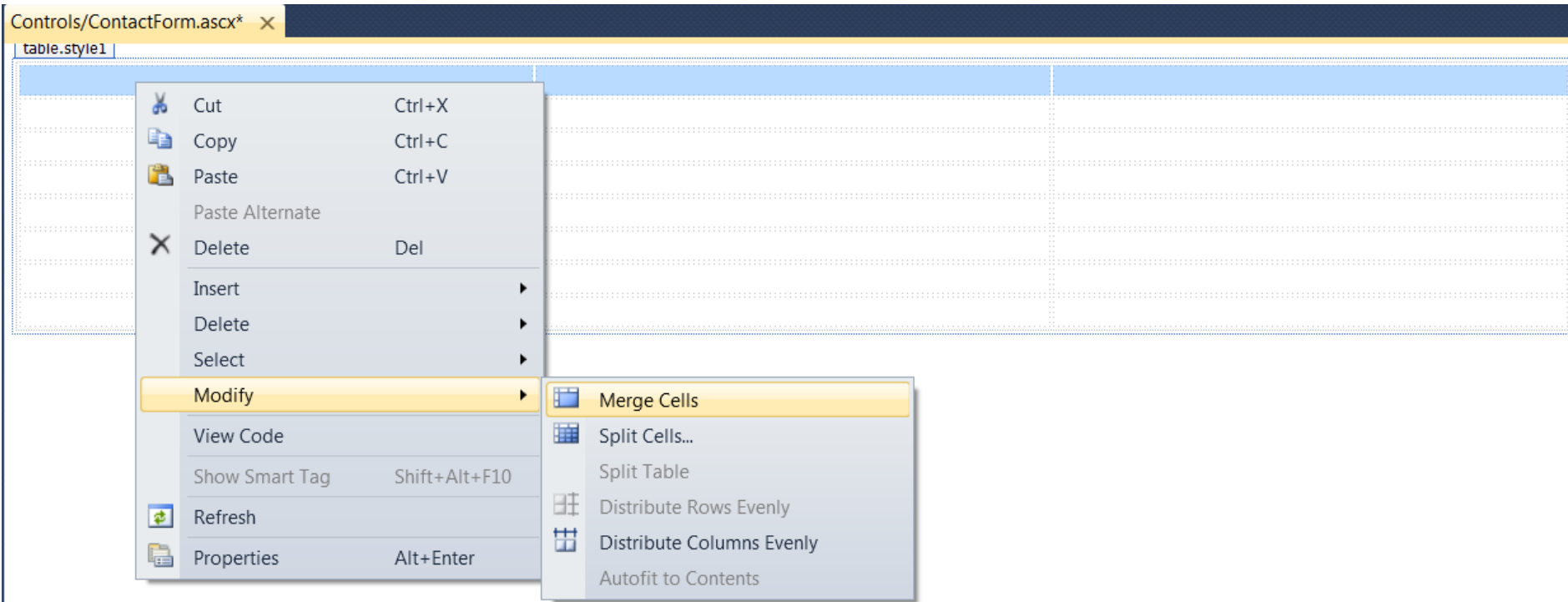
# Example

- In Design View → Insert a table



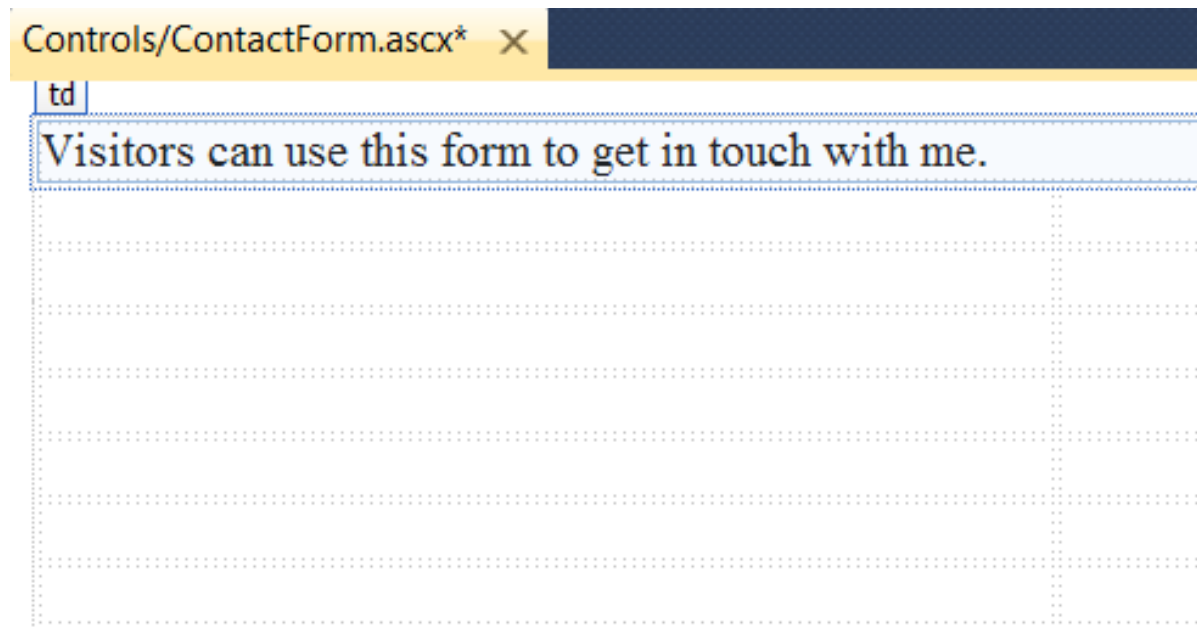
# Example

- Merge the 3 cells of the 1<sup>st</sup> row.



# Example

- Type some text.



Controls/ContactForm.ascx\* X

td

Visitors can use this form to get in touch with me.

# Example

- Type **Name** → 1<sup>st</sup> cell of the 2<sup>nd</sup> row
- Drag a **TextBox** → 2<sup>nd</sup> cell of the 2<sup>nd</sup> row
- Drag a **RequiredFieldValidator** → 3<sup>rd</sup> cell of the 2<sup>nd</sup> row
- Drag a **Button** → 2<sup>nd</sup> cell of the last row

Controls/ContactForm.ascx\* ×

Visitors can use this form to get in touch with me.

Name

RequiredFieldValidator

Button

# Example

- **TextBox** → **ID = nameTextBox**
- **Button:**
  - **ID = sendButton**
  - **Text = Send**

# Example

- Properties of `RequiredFieldValidator`:
  - `CssClass = ErrorMessage`
  - `ErrorMessage = Enter your name`
  - `Text = * Enter your name`
  - `ControlToValidate = nameTextBox`

# Example

- Add the following declaration to **Monochrome.css** and **DarkGrey.css**.

```
.ErrorMessage  
{  
    color:Red;  
}
```

# Example

- Open **Contact.aspx** from the **About** folder.
- In Design View:
  - From the Solution Explorer, drag the user control **ContactForm.ascx** into the main content area of the page, identified by the purple border.



[Home](#) [Reviews](#) [About](#) [Login](#)

[Home](#)

[Reviews](#)

[By Genre](#)

[All Reviews](#)

[About](#)

[Contact](#)

[About Me](#)

[Login](#)

SiteMapDataSource - SiteMapDataSource1

[Root Node](#) > [Parent Node](#) > Current Node

cpMainContent (Custom)

Select a theme

Monochrome



[Home](#) [Reviews](#) [About](#) [Login](#)

[Home](#)

[Reviews](#)

[By Genre](#)

[All Reviews](#)

[About](#)

[Contact](#)

[About Me](#)

[Login](#)

SiteMapDataSource - SiteMapDataSource1

[Root Node](#) > [Parent Node](#) > Current Node

uc1:contactform#ContactForm1

Visitors can use this form to get in touch with me.

Name

\* Enter your name

Send

Select a theme

Monochrome

# Example

- In Source View:

```
<asp:Content ID="Content2" ContentPlaceHolderID="cpMainContent" Runat="Server">  
    <uc1:ContactForm ID="ContactForm1" runat="server" />  
</asp:Content>
```

# Example

- View in browser

[Home](#) > [About](#) > Contact

Visitors can use this form to get in touch with me.

Name

# Example

- Leave the **TextBox** empty and click **Send**

[Home](#) > [About](#) > Contact

Visitors can use this form to get in touch with me.

Name

\*  
Enter  
your  
name

# Example

- Enter your name & click Send
  - Page posts back to the server

[Home](#) > [About](#) > Contact

Visitors can use this form to get in touch with me.

Name

# Example

# Extending the Contact Form

- Open **ContactForm.ascx** in **Controls** folder.
- In Design View → In the 2<sup>nd</sup> column, drag 5 **TextBox** in the empty table cells between the text box for the **Name** and the **Send** button.

[illegible]

# Example

## Extending the Contact Form

- Name the IDs:
  - `emailAddressTextBox`
  - `confirmEmailAddressTextBox`
  - `phoneHomeTextBox`
  - `phoneBusinessTextBox`
  - `commentsTextBox`
    - `TextMode = MultiLine`

# Example

## Extending the Contact Form

Visitors can use this form to get in touch with me.

Name	<input type="text"/>	* Enter your name
	<input type="text"/>	
	<input type="text"/>	
	<input type="text"/>	
	<input type="text"/>	
<div><div></div><div></div></div>		
<div>Send</div>		



# Example

## Extending the Contact Form

- Add the text:

Visitors can use this form to get in touch with me.	
Name	<input type="text"/>
Email Address	<input type="text"/>
Email Address (Repeat)	<input type="text"/>
Home Phone	<input type="text"/>
Business Phone	<input type="text"/>
Comments	<div><div></div><div></div></div>
<input type="button" value="Send"/>	
* Enter your name	

# Example

## Extending the Contact Form

- In the last cell of the row for the first e-mail address, drag a **RequiredFieldValidator** and a **RegularExpressionValidator**.
- In the last cell of the row for the second e-mail address, drag a **RequiredFieldValidator** and a **CompareValidator**.
- In the last cell for the comments row, drag a **RequiredFieldValidator**.

# Example

## Extending the Contact Form

Visitors can use this form to get in touch with me.	
Name	<input type="text"/>
Email Address	<input type="text"/>
Email Address (Repeat)	<input type="text"/>
Home Phone	<input type="text"/>
Business Phone	<input type="text"/>
Comments	<div><div></div><div></div></div>
<input type="button" value="Send"/>	

\* Enter your name

RequiredFieldValidatorRegularExpressionValidator

RequiredFieldValidator CompareValidator

RequiredFieldValidator

# Example

## Extending the Contact Form

- For each of the 5 validation controls set the properties:
  - `Text = *`
  - `Display = Dynamic`
  - `CssClass = ErrorMessage`

# Example

## Extending the Contact Form

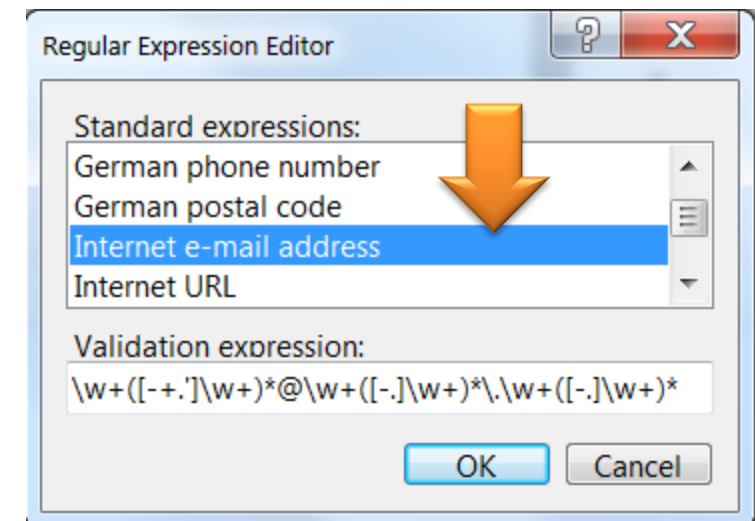
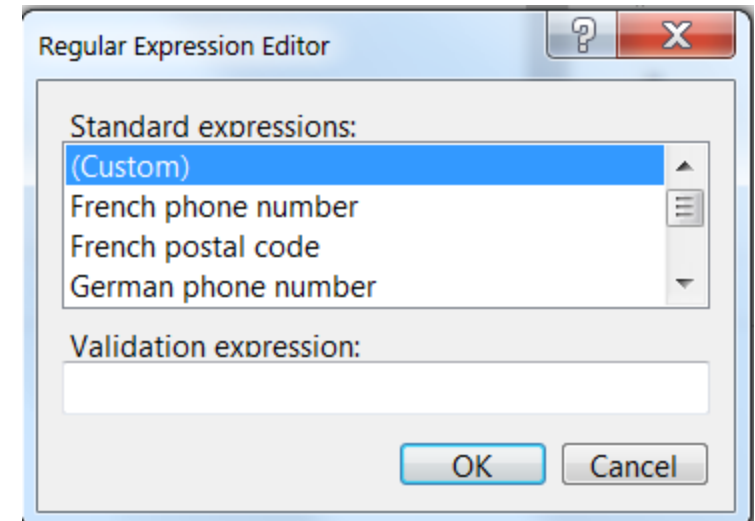
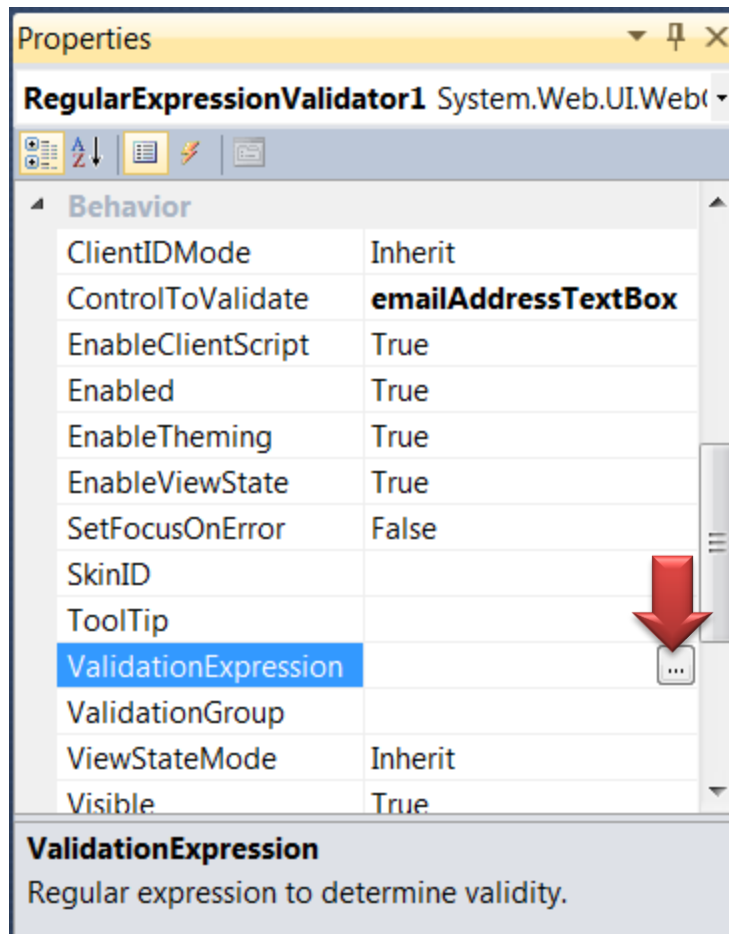
- Set the remaining properties for the controls as shown:

CONTROL	PROPERTIES YOU NEED TO SET	
RequiredFieldValidator (for the first e-mail address)	ErrorMessage:	Enter an e-mail address
	ControlToValidate:	emailAddressTextBox
RegularExpressionValidator	ErrorMessage:	Enter a valid e-mail address
	ControlToValidate:	emailAddressTextBox
RequiredFieldValidator (for the second e-mail address)	ErrorMessage:	Confirm the e-mail address
	ControlToValidate:	confirmEmailAddressTextBox
CompareValidator	ErrorMessage:	Retype the e-mail address
	ControlToCompare:	emailAddressTextBox
	ControlToValidate:	confirmEmailAddressTextBox
RequiredFieldValidator (for the Comments field)	ErrorMessage:	Enter a comment
	ControlToValidate:	commentsTextBox

# Example

## Extending the Contact Form

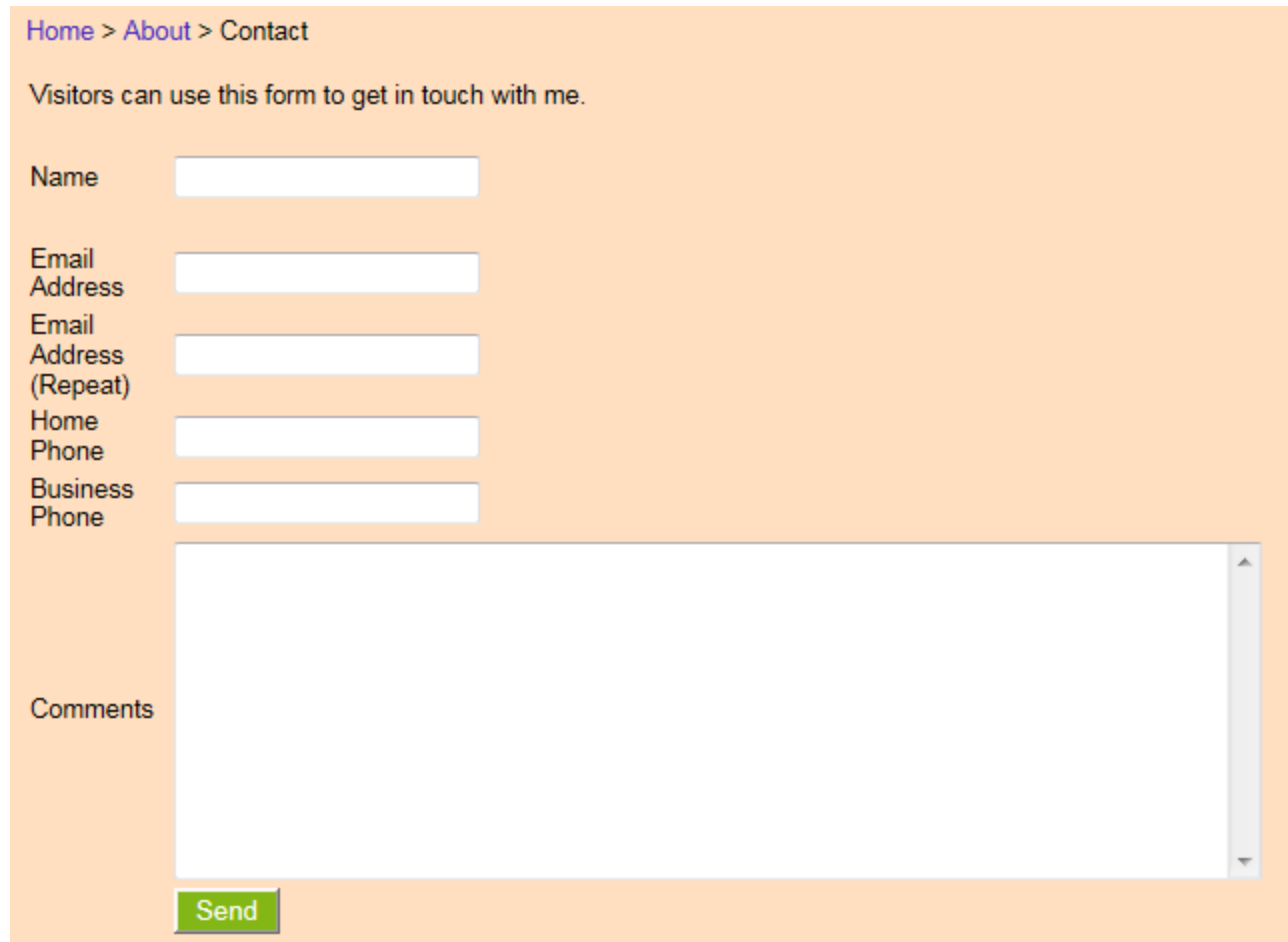
- Click **RegularExpressionValidator**



# Example

## Extending the Contact Form

- View `Contact.aspx` in browser



The screenshot shows a web page with an orange background. At the top, there is a breadcrumb navigation link: [Home](#) > [About](#) > [Contact](#). Below this, a message reads: "Visitors can use this form to get in touch with me." The form contains several input fields: "Name", "Email Address", "Email Address (Repeat)", "Home Phone", and "Business Phone". Each field is represented by a white rectangular box. Below these fields is a large text area for "Comments", which has a vertical scrollbar on its right side. At the bottom left of the form, there is a green button with the text "Send".

# Example

## Extending the Contact Form

- Leave textboxes empty and click **Send**

[Home](#) > [About](#) > Contact

Visitors can use this form to get in touch with me.

Name

Email Address

Email Address (Repeat)

Home Phone

Business Phone

Comments

\* Enter your name

\*

\*

\*



# Validation Methods

- Open → **ContactForm.ascx**
- In Design View → Insert a row below the last row.
  - **Modify > Merge Cells**

Visitors can use this form to get in touch with me.	
Name	<input type="text"/>
Email Address	<input type="text"/>
Email Address (Repeat)	<input type="text"/>
Home Phone	<input type="text"/>
Business Phone	<input type="text"/>
Comments	<div><div></div><div></div></div>
td.style2	<div>Send</div>

# Example

## Validation Methods

- Drag a **ValidationSummary** control into this new cell
  - **CssClass = ErrorMessage**

Visitors can use this form to get in touch with me.

Name	<input type="text"/>	* Enter your name
Email Address	<input type="text"/>	**
Email Address (Repeat)	<input type="text"/>	**
Home Phone	<input type="text"/>	
Business Phone	<input type="text"/>	
Comments	<div><div></div><div></div></div>	*

asp:validationsum...#ValidationSum...

- Error message 1.
- Error message 2.

# Example

## Validation Methods

- In the empty cell after the text box for the **Home Phone**, drag a **CustomValidator** control.

Visitors can use this form to get in touch with me.

Name	<input type="text"/>	* Enter your name
Email Address	<input type="text"/>	**
Email Address (Repeat)	<input type="text"/>	** asp:customvalidator#CustomValidator1
Home Phone	<input type="text"/>	CustomValidator
Business Phone	<input type="text"/>	
Comments	<input type="text"/>	*
<input type="button" value="Send"/>		
<ul style="list-style-type: none"><li>• Error message 1.</li><li>• Error message 2.</li></ul>		

# Example

## Validation Methods

- Set the following properties:
  - `CssClass = ErrorMessage`
  - `Display = Dynamic`
  - `ErrorMessage = Enter your home or business number`
  - `Text = *`
  - `ClientValidationFunction = ValidatePhoneNumbers`

# Example

## Validation Methods

- Double-click CustomValidator

```
protected void CustomValidator1_ServerValidate(object source, ServerValidateEventArgs args)
{
    if (!string.IsNullOrEmpty(phoneHomeTextBox.Text) || !string.IsNullOrEmpty(phoneBusinessTextBox.Text))
    {
        args.IsValid = true;
    }
    else
    {
        args.IsValid = false;
    }
}
```

**Server-side Validation**

# Example

## Validation Methods

- In Source View:

```
<script type="text/javascript">
    function ValidatePhoneNumbers(source, args)
    {
        var phoneHome = document.getElementById('<%= phoneHomeTextBox.ClientID %>');
        var phoneBusiness = document.getElementById('<%= phoneBusinessTextBox.ClientID %>');
        if (phoneHome.value != '' || phoneBusiness.value != '')
        {
            args.IsValid = true;
        }
        else
        {
            args.IsValid = false;
        }
    }
</script>
```

```
<table class="style1">
    <tr>
        <td colspan="3">
            Visitors can use this form to get in touch with me.</td>
        ..
    </tr>
</table>
```

# Example

## Validation Methods

[Home](#) > [About](#) > [Contact](#)

Visitors can use this form to get in touch with me.

Name

Email Address

Email Address (Repeat)

Home Phone

Business Phone

Comments

- Enter a valid e-mail address
- Confirm the e-mail address
- Enter your home or business phone number
- Enter a comment

\*

\*

\*

\*

View  
**Contact.aspx**  
in  
browser.

# Example

## Validation Methods

- In Design View:
  - **ValidationSummary** control's Properties Grid
    - **ShowMessageBox = True**
    - **ShowSummary = False**
    - **HeaderText = Please correct the following errors before you press the Send Button:**



# Example

## Validation Methods

View  
**Contact.aspx**  
in  
browser.

Home > About > Contact

Visitors can use this form to get in touch with me.

Name

Email Address

Email Address (Repeat)

Home Phone


Business Phone

Comments

\* Enter your name

\*

Message from webpage

 Please correct the following errors before you press the Send Button:

- Enter your name
- Enter an e-mail address
- Confirm the e-mail address
- Enter your home or business phone number
- Enter a comment

# Sending E-Mail

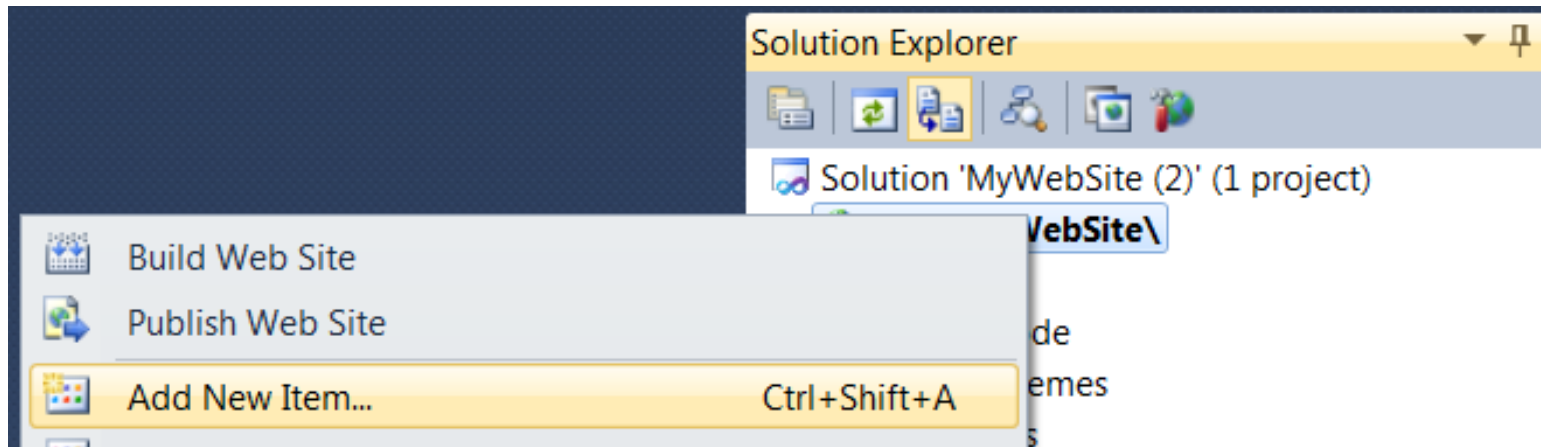
- Sending e-mail from .NET application:

CLASS	DESCRIPTION
<code>MailMessage</code>	This class represents the message you're going to send. It has properties such as <code>Subject</code> and <code>Body</code> to set the message contents; <code>To</code> , <code>CC</code> , and <code>Bcc</code> properties to set the addressees; and an <code>Attachments</code> collection to attach files to the message.
<code>MailAddress</code>	This class represents a sender or receiver address used in the e-mail. It has a few constructor overloads that enable you to set the e-mail address and display name.
<code>Attachment</code>	This class represents files you can attach to a <code>MailMessage</code> . When you construct an <code>Attachment</code> instance, you can pass in the name of the file you want to send. You then add the attachment to the <code>MailMessage</code> using the <code>Add</code> method of its <code>Attachments</code> collection.
<code>SmtpClient</code>	This class is used to send the actual message. By default, an instance of this class checks the <code>web.config</code> file for settings such as the SMTP server (which stands for Simple Mail Transfer Protocol) to send the mail to and an optional user name and password that is used for sending e-mail.

# Example

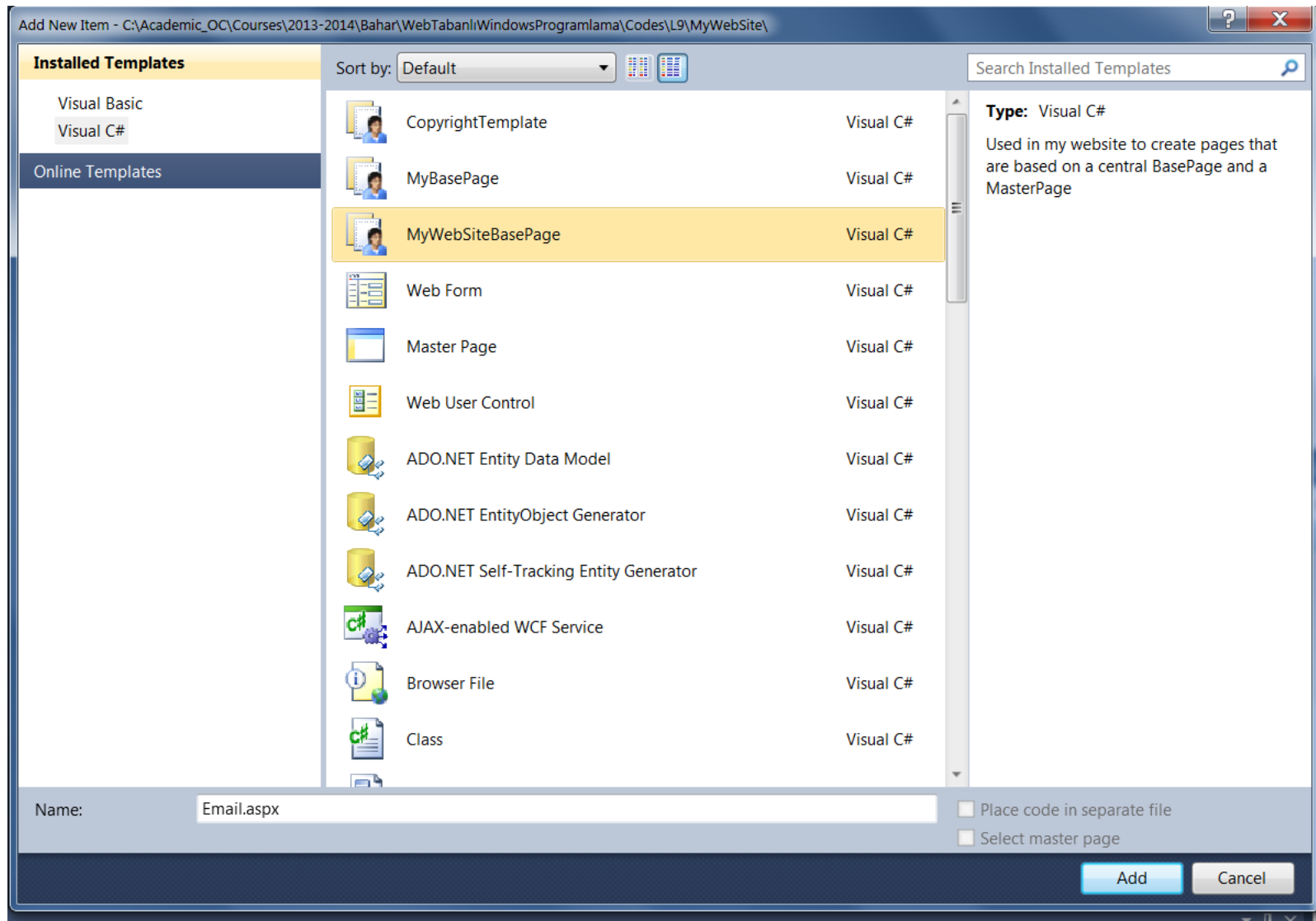
## Sending E-Mail

- Add New Item



# Example

## Sending E-Mail



# Example

## Sending E-Mail

- Change the **Title** → **Email Demo**
- Open Code Behind file
  - Add:

```
using System.Net.Mail;
```

# Example

## Sending E-Mail

- Add the following code to **Page\_Load** handler:

```
protected void Page_Load(object sender, EventArgs e)
{
    MailMessage myMessage = new MailMessage();
    myMessage.Subject = "Test Message";
    myMessage.Body = "Hello world! This is my web site.";
    myMessage.From = new MailAddress("wbwpcourse@gmail.com", "Sender_WBWP");
    myMessage.To.Add(new MailAddress("wbwpcourse@gmail.com", "Receiver_Ozgu Can"));
    myMessage.CC.Add(new MailAddress("ozgucan@gmail.com", "Receiver_Ozgu Can"));

    SmtpClient mySmtpClient = new SmtpClient();
    mySmtpClient.Host = "smtp.gmail.com";
    mySmtpClient.EnableSsl = true;
    mySmtpClient.Port = 587;
    mySmtpClient.Send(myMessage);
}
```

# Example

## Sending E-Mail

- Open `web.config`
- Right before the closing `</configuration>` tag, add the following settings:

```
<system.net>  
  <mailSettings>  
    <smtp deliveryMethod="Network" from="WBWP < wbwpcourse@gmail.com >">  
      <network enableSsl="true" userName="wbwpcourse@gmail.com" password="blablabla" host="smtp.google.com" />  
    </smtp>  
  </mailSettings>  
</system.net>  
</configuration>
```

# Example

## Sending E-Mail

- View **Email.aspx** in browser
- Check your e-mail box!



# Reading from Text Files

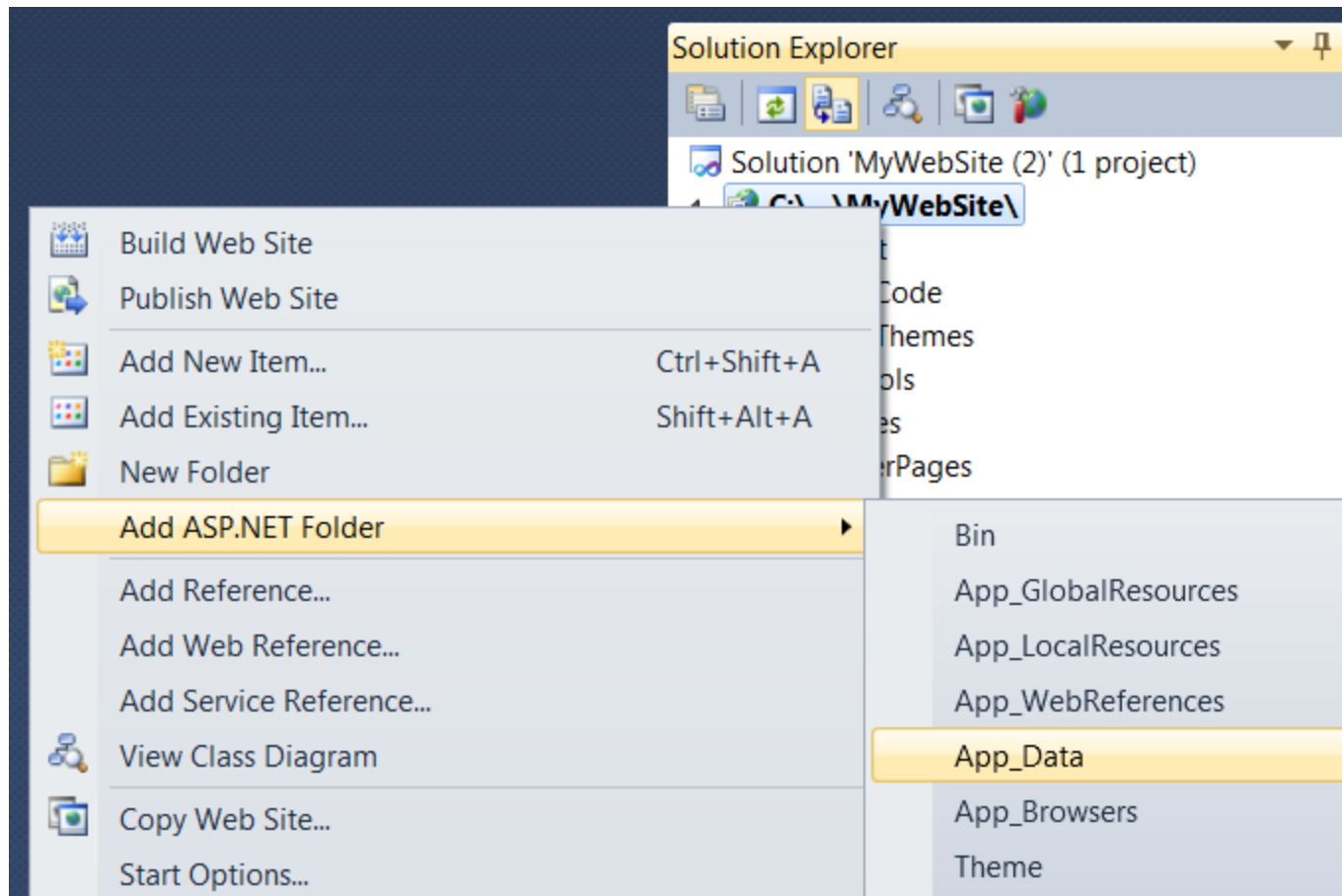
- Common methods of the **File** class

METHOD	VALUE
AppendAllText	Appends a specified string to a text file. If the file does not exist, it's created first.
Copy	Copies a file from one location to another.
Delete	Deletes the specified file from disk.
Exists	Checks if the specified file exists on disk.
Move	Moves the specified file to a different location.
ReadAllText	Reads the contents of a text file.
WriteAllText	Writes the contents of a string to a new file and overwrites the target file if it already exists.

# Example

## Sending E-Mail from the ContactForm

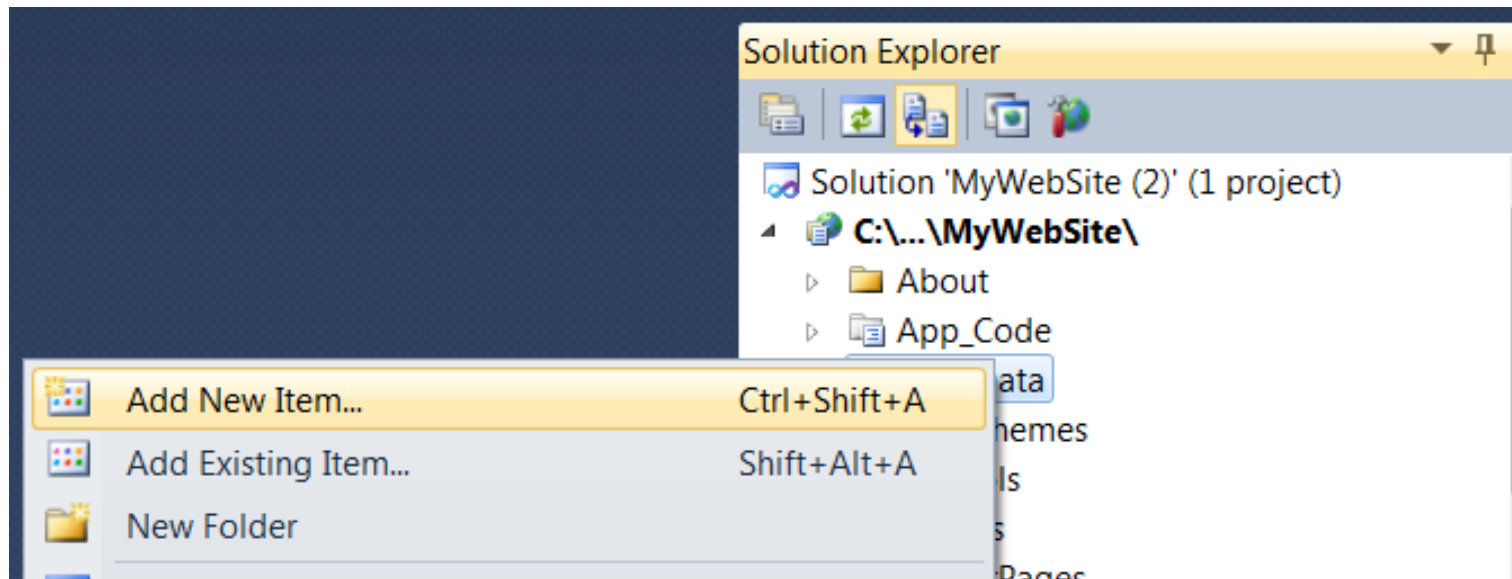
- Add new folder → **App\_Data**



# Example

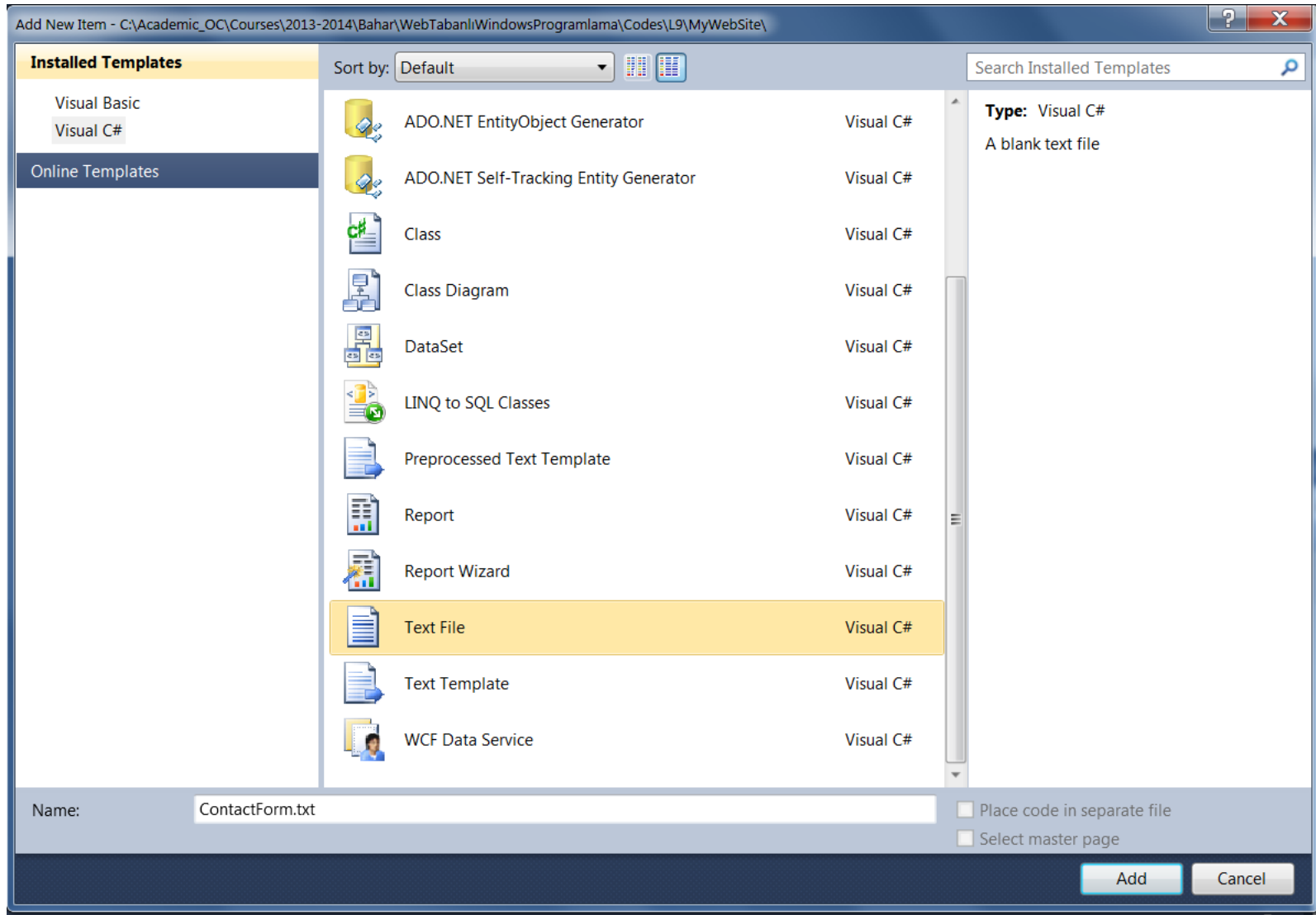
## Sending E-Mail from the ContactForm

- Add a new text file to **App\_Data** folder



# Example

## Sending E-Mail from the ContactForm



# Example

## Sending E-Mail from the ContactForm

- Enter the following text:

App\_Data/ContactForm.txt\* X

Hi there,

A user has left the following feedback at the site:

Name: ##Name##

E-mail address: ##Email##

Home phone: ##HomePhone##

Business phone: ##BusinessPhone##

Comments: ##Comments##

# Example

## Sending E-Mail from the ContactForm

Open code behind file → **ContactForm.ascx.cs**

Add the following namespaces:

```
using System.IO; // Provides access to the File class for reading the file
using System.Net.Mail; // Provides access to the various mail related classes
```

# Example

## Sending E-Mail from the ContactForm

- In Source View:
  - To hide the entire table programmatically when the form has been submitted:

```
<table class="style1">
```



```
<table class="style1" runat="server" id="FormTable">
```

# Example

## Sending E-Mail from the ContactForm

- After `</table>` tag:

```
</table>
```

```
<asp:Label ID="Message" runat="server" Text="Message Sent!" Visible="False"></asp:Label>
```



# Example

## Sending E-Mail from the ContactForm

- In Design View:
  - **ValidationSummary** control's properties:
    - **ShowSummary = True**
    - **ShowMessageBox = False**

# Double-click **Send** button

```
protected void sendButton_Click(object sender, EventArgs e)
{
    if (Page.IsValid)
    {
        string fileName = Server.MapPath("~/App_Data/ContactForm.txt");
        string mailBody = File.ReadAllText(fileName);

        mailBody = mailBody.Replace("##Name##", nameTextBox.Text);
        mailBody = mailBody.Replace("##Email##", emailAddressTextBox.Text);
        mailBody = mailBody.Replace("##HomePhone##", phoneHomeTextBox.Text);
        mailBody = mailBody.Replace("##BusinessPhone##", phoneBusinessTextBox.Text);
        mailBody = mailBody.Replace("##Comments##", commentsTextBox.Text);

        MailMessage myMessage = new MailMessage();
        myMessage.Subject = "Response from web site";
        myMessage.Body = mailBody;
        myMessage.From = new MailAddress("wbwpcourse@gmail.com", "Sender Name");
        myMessage.To.Add(new MailAddress("wbwpcourse@gmail.com", "Receiver Name"));

        SmtpClient mySmtpClient = new SmtpClient();
        mySmtpClient.Host = "smtp.gmail.com";
        mySmtpClient.EnableSsl = true;
        mySmtpClient.Port = 587;
        mySmtpClient.Send(myMessage);

        Message.Visible = true;
        FormTable.Visible = false;
        System.Threading.Thread.Sleep(5000);
    }
}
```

# Example

## Sending E-Mail from the ContactForm

- View `Contact.aspx` in browser

[Home](#) > [About](#) > [Contact](#)

Visitors can use this form to get in touch with me.

Name

Email Address

Email Address (Repeat)

Home Phone

Business Phone

Comments

[Home](#) > [About](#) > [Contact](#)

Message Sent!



# Read

- **specifiedPickupDirectory**
- Sets the directory where applications save mail messages to be processed by the SMTP server.

<http://msdn.microsoft.com/en-us/library/ms164241.aspx>