

ASP.NET

.NET support for Internet Applications

ASP.NET

ASP.NET comprises two technologies:

■ Web Forms

- For creating web browser hosted GUI applications
- An evolution from Microsoft's original Active Server Pages (ASP)

■ Web Services

- For exposing and accessing functionality on remote machines.
- Don't provide a GUI; more like remote procedure calls.
- Uses open industry "standards" such as XML and SOAP.
 - Works across platforms.
- Covered in Week 9 Lecture.

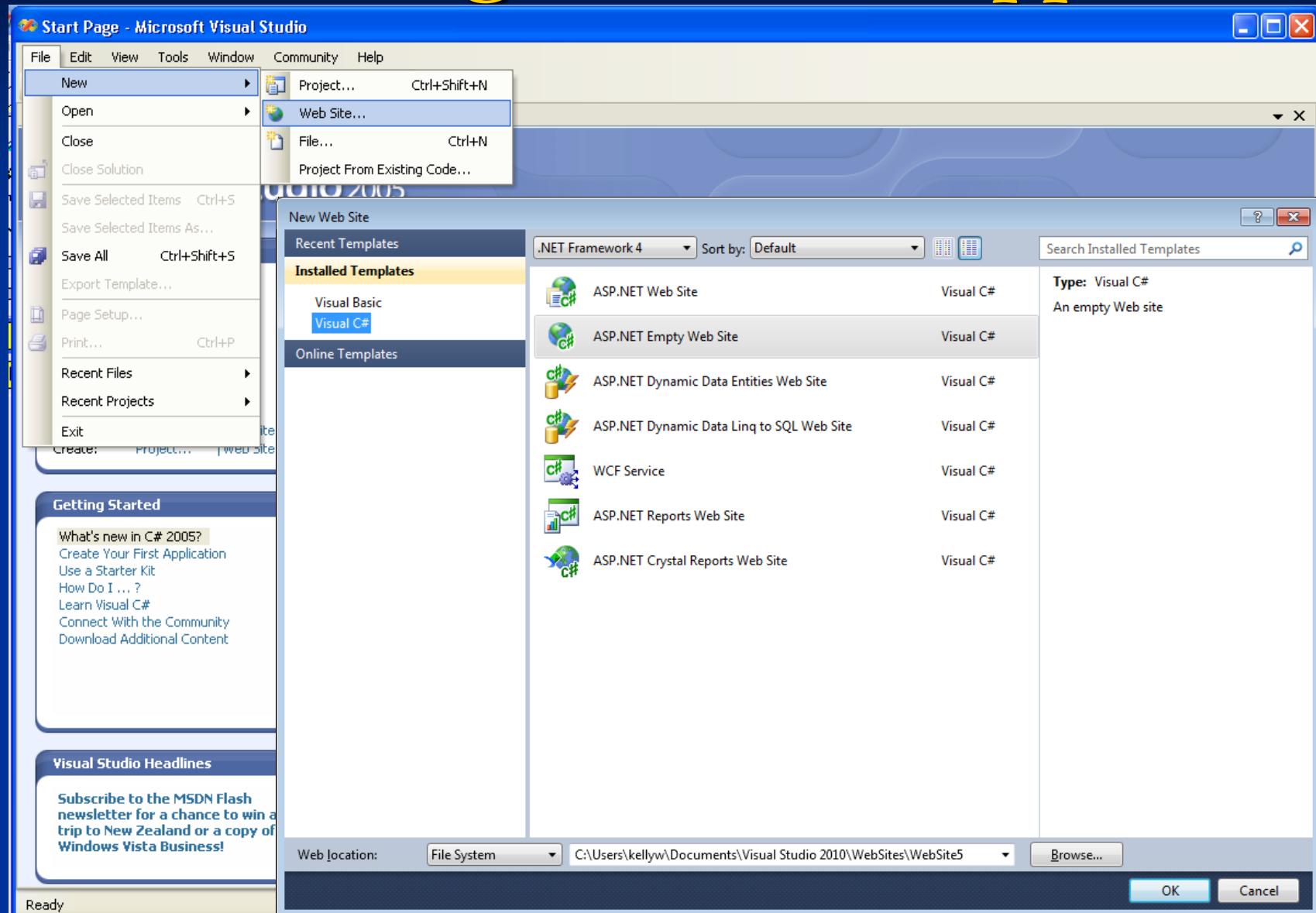
Traditional ASP Example

```
<%@ language = "vbscript" %>
<HTML>
<HEAD>
<TITLE>Simple Form Processing</TITLE>
</HEAD>
<BODY>
<h1>Let us get to know you</h1>
<%
if Request.ServerVariables("CONTENT_LENGTH") = 0 then
%>
    <h3>Please submit the form</h3>
    <form name = form1 action = /asp/time.asp method = post ID="Form1">
        Name: <input type = text name = name ID="Text1"><br>
        Likes:
        <input type = radio name = like checked value = hindi ID="Radio1">Hindi Movies
        <input type = radio name = like value = English ID="Radio2">English Movies
        <br>
        <input type = submit name = submit1 value = Send ID="Submit1">
        <input type = reset value = ResetAll ID="Reset1" NAME="Reset1">
    </form>
<%
else
    Response.write "I think I know you " & Request.form("name") & "<br>"
    Response.write "Even I like " & Request.form("like")
%>
<%
end if
%>
</BODY>
</HTML>
```

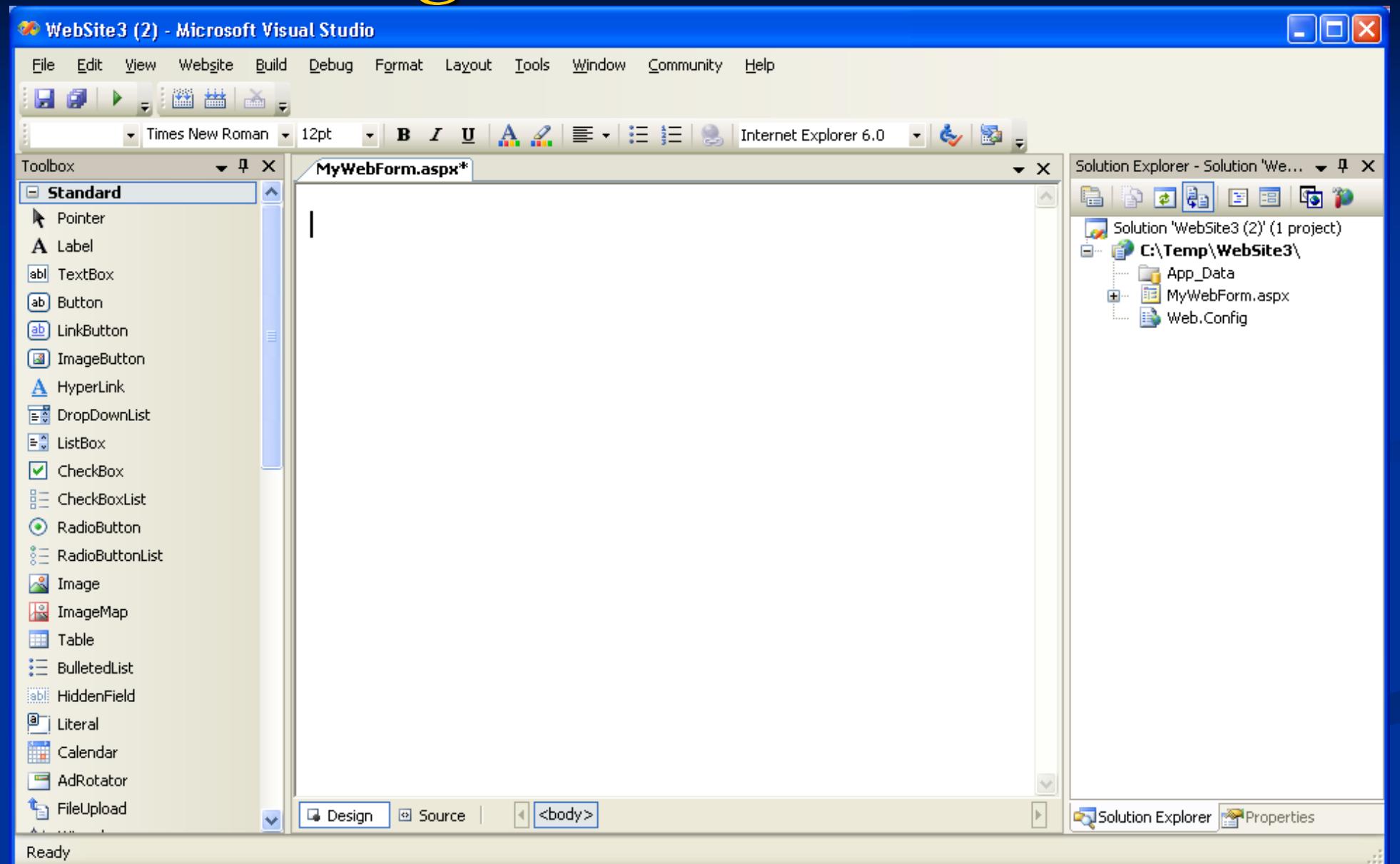
ASP.NET Features Overview

- Separates user interface from code:
 - User interface (.aspx file)
 - *Code behind* file (any .NET language)
- Server-side code is JIT compiled by CLR to native code.
- New server-side event model.
- Greater variety of control types
 - Web controls
 - Programmers can create their own reusable web controls.
- ASP.NET tags are processed and translated into standard HTML before returning the response to the client's web browser.
 - Clients can use any platform or web browser.
- Supports WYSIWYG design of Web Forms

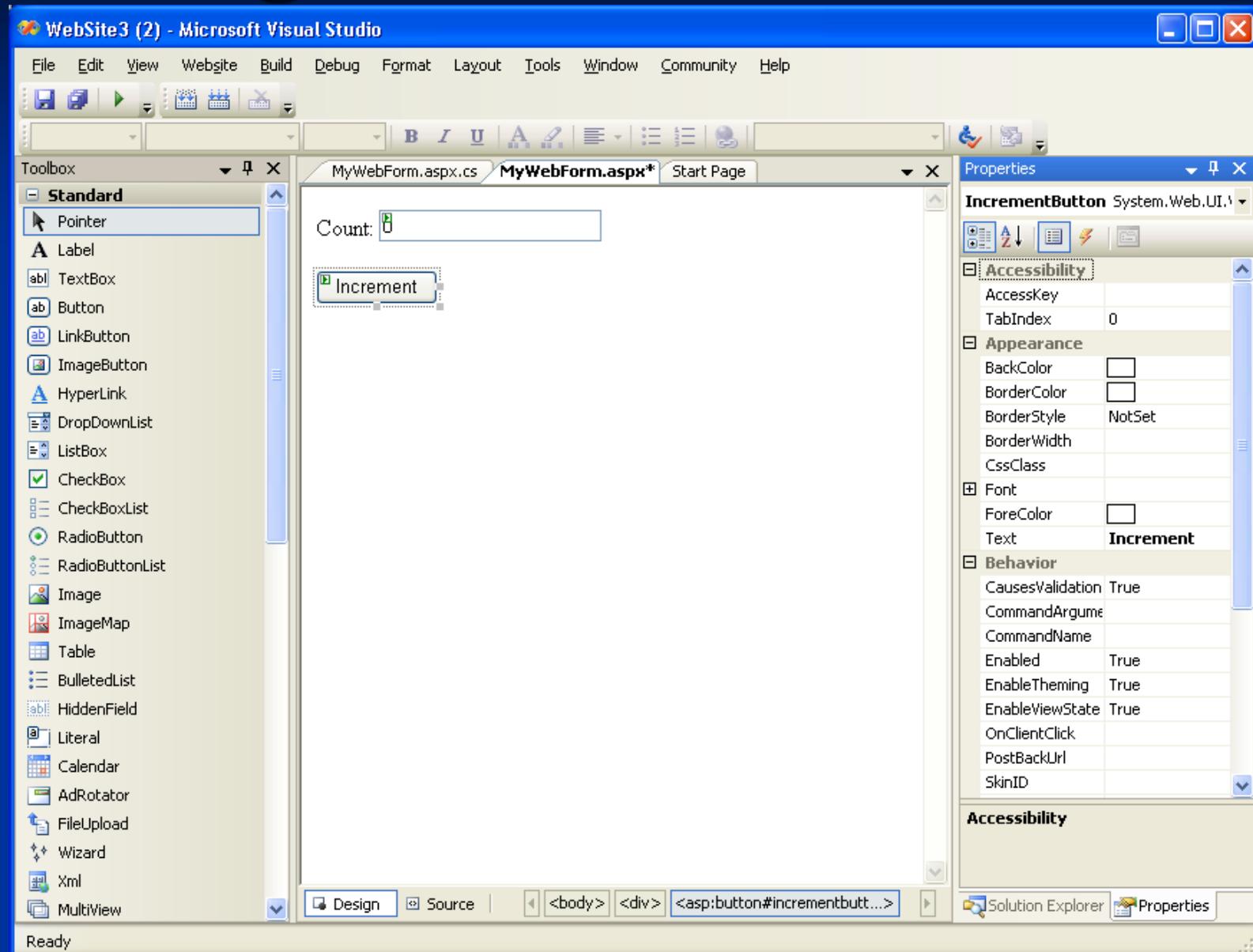
Creating ASP.NET Applications



Visual Design of ASP.NET Web Forms



Adding ASP.NET Web Controls



MyWebForm.aspx

The screenshot shows the Microsoft Visual Studio interface with the title bar "Diary - Microsoft Visual Studio". The menu bar includes FILE, EDIT, VIEW, WEBSITE, BUILD, DEBUG, TEAM, SQL, FORMAT, TOOLS, TEST, ARCHITECTURE, and ANALYZE. The toolbar has icons for back, forward, search, and other development tools. The status bar at the bottom shows "Ready", "Ln 24", "Col 8", "Ch 8", and "INS". The code editor window displays the ASPX page "MyWebForm.aspx". The code includes an ASPX header, an HTML document structure, and a form containing a text box and a button.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="MyWebForm.aspx.cs" Inherits="MyWebForm" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title> My Web Form </title>
    <style type="text/css">
        #form1 {
            height: 200px;
            width: 600px;
        }
    </style>
</head>
<body style="height: 190px; width: 681px">
    <form id="form1" runat="server">
        Count:
        <asp:TextBox ID="Count" runat="server"> 0 </asp:TextBox>
        <p></p>
        <asp:Button ID="IncrementButton" runat="server" Text="Increment" />
    </form>
</body>
</html>
```

Adding Standard HTML Controls

The screenshot shows the Microsoft Visual Studio interface for a web application named "WebSite3 (2)". The main window displays the design view of a page named "MyWebForm.aspx". The page contains a text input field with the placeholder "Count: 0" and two buttons: "Increment" and "Other". The "Properties" panel on the right shows the properties for the "Increment" button, which is identified as "Button1". The "Misc" section of the properties panel includes the following settings:

(Id)	Button1
AtomicSelection	
Class	
ContentEditable	inherit
Dir	
Disabled	
HideFocus	
Lang	
Language	
Name	
RunAt	
Size	20
Style	
Title	
Type	button

The "Toolbox" on the left lists various standard and data controls, with "HTML" expanded to show items like Pointer, Input (Button), Input (Reset), etc.

MyWebForm.aspx

The screenshot shows the Microsoft Visual Studio interface with the title bar "Diary - Microsoft Visual Studio". The menu bar includes FILE, EDIT, VIEW, WEBSITE, BUILD, DEBUG, TEAM, SQL, FORMAT, TOOLS, TEST, ARCHITECTURE, and ANALYZE. The toolbar has icons for back, forward, search, and file operations. The status bar at the bottom shows "Item(s) Saved", "Ln 22", "Col 1", "Ch 1", and "INS". The main window displays the ASPX code for "MyWebForm.aspx". The code includes an @Page directive, DOCTYPE declaration, HTML and head tags, a style block for form1, and a body tag containing a form with a TextBox, a Button labeled "Increment", a paragraph, and another Button labeled "Other". The "Source" tab is selected, and the code is displayed in a syntax-highlighted editor.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="MyWebForm.aspx.cs" Inherits="MyWebForm" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title> My Web Form </title>
    <style type="text/css">
        #form1 {
            height: 200px;
            width: 600px;
        }
    </style>
</head>
<body style="height: 190px; width: 681px">
    <form id="form1" runat="server">
        Count:
        <asp:TextBox ID="Count" runat="server"> 0 </asp:TextBox>
        <p></p>
        <asp:Button ID="IncrementButton" runat="server" Text="Increment" />
        <p></p>
        <asp:Button ID="Button1" runat="server" Text="Other" />
    </form>
</body>
</html>
```

Web Form containing ASP.NET tags

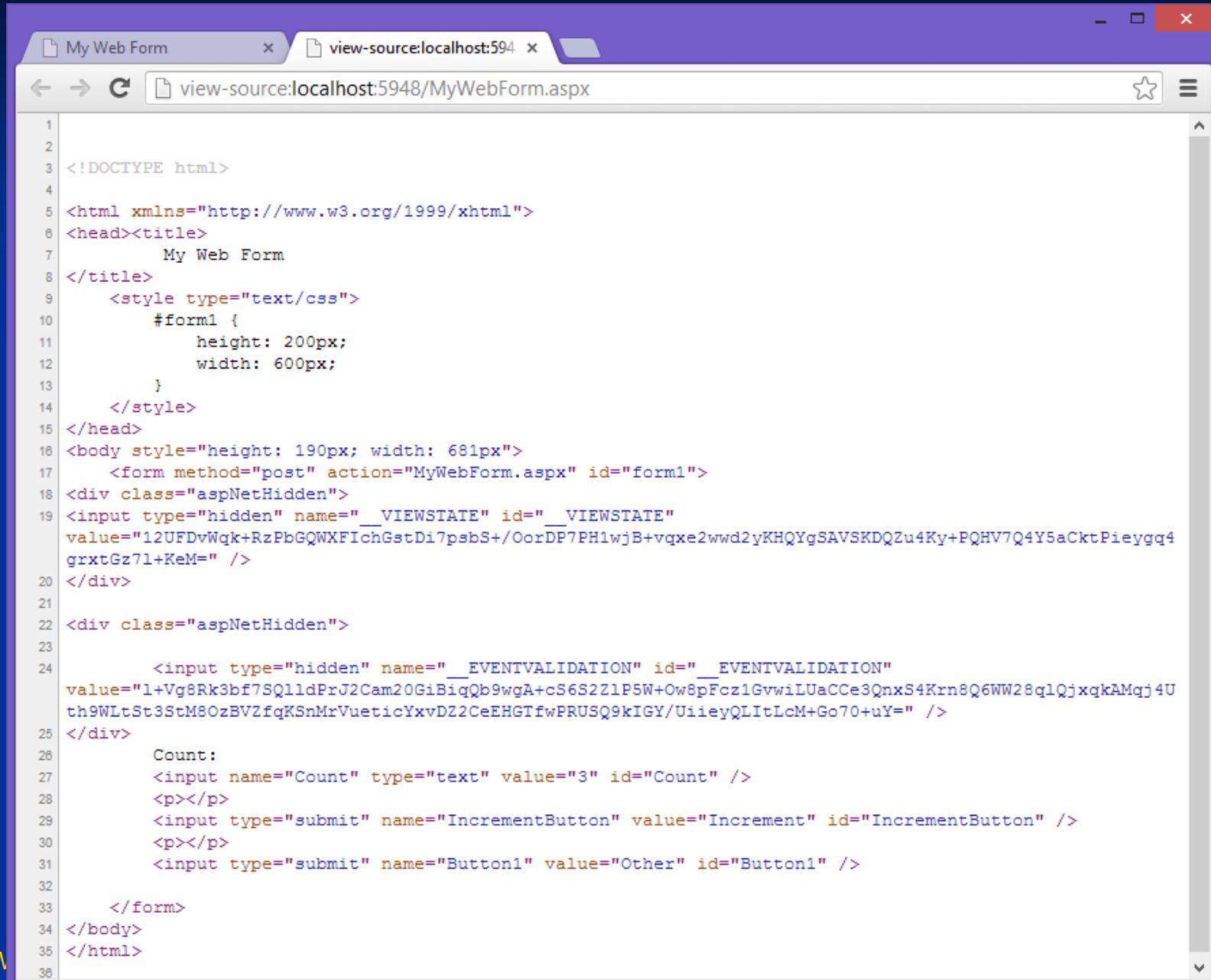
The screenshot shows the Microsoft Visual Studio interface with the title bar "Diary - Microsoft Visual Studio". The menu bar includes FILE, EDIT, VIEW, WEBSITE, BUILD, DEBUG, TEAM, SQL, FORMAT, TOOLS, TEST, ARCHITECTURE, and ANALYZE. The toolbar has icons for file operations like Open, Save, Print, and a search icon. The status bar at the bottom shows "Item(s) Saved", "Ln 25", "Col 8", "Ch 8", and "INS". The main code editor window displays the ASPX page "MyWebForm.aspx". The code includes the following structure:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="MyWebForm.aspx.cs" Inherits="MyWebForm" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
  <head runat="server">
    <title> My Web Form </title>
    <style type="text/css">
      #form1 {
        height: 200px;
        width: 600px;
      }
    </style>
  </head>
  <body style="height: 190px; width: 681px">
    <form id="form1" runat="server">
      Count:
      <asp:TextBox ID="Count" runat="server"> 0 </asp:TextBox>
      <p></p>
      <asp:Button ID="IncrementButton" runat="server" Text="Increment" />
      <p></p>
      <asp:Button ID="Button1" runat="server" Text="Other" />
    </form>
  </body>
</html>
```

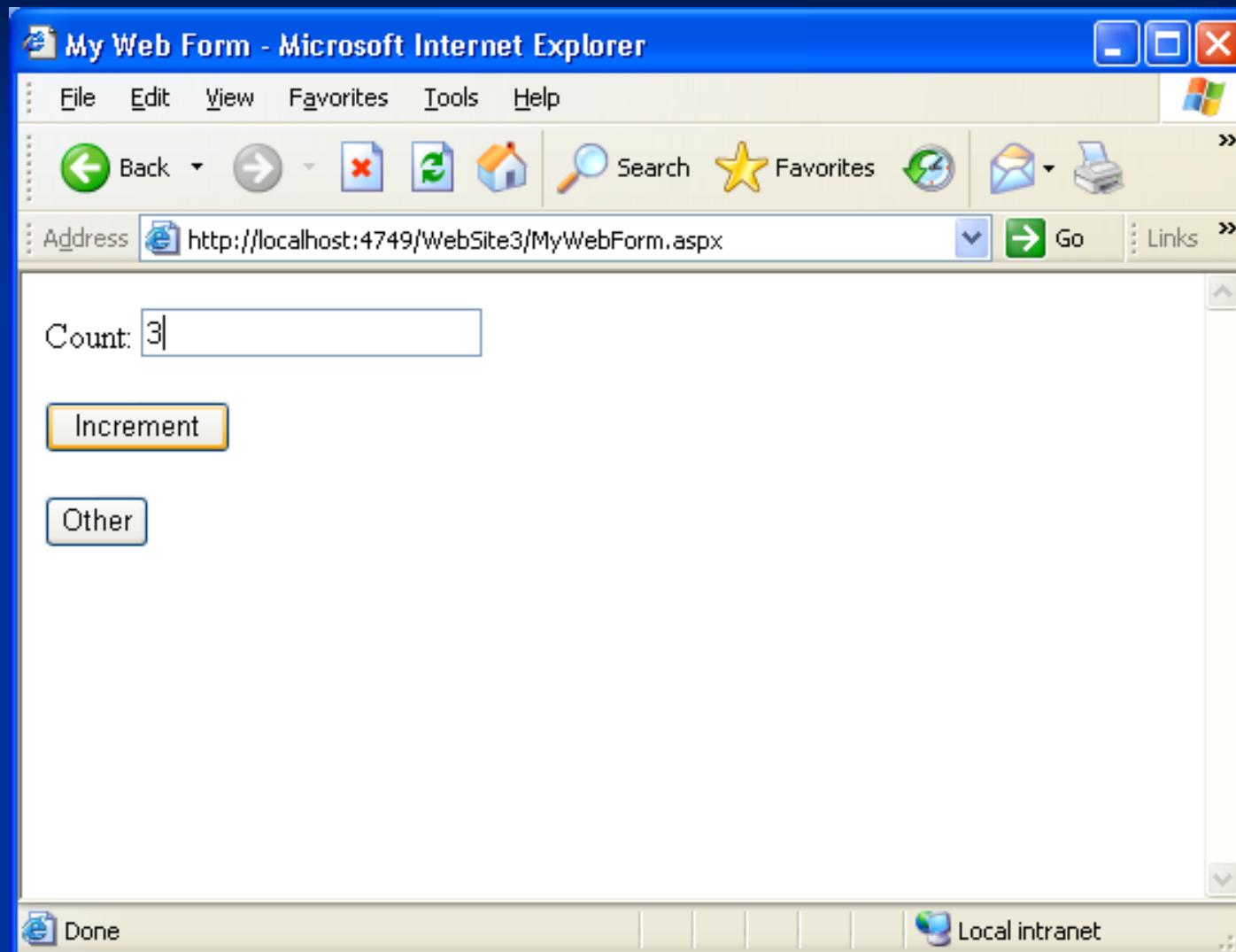
Pure HTML sent to Client



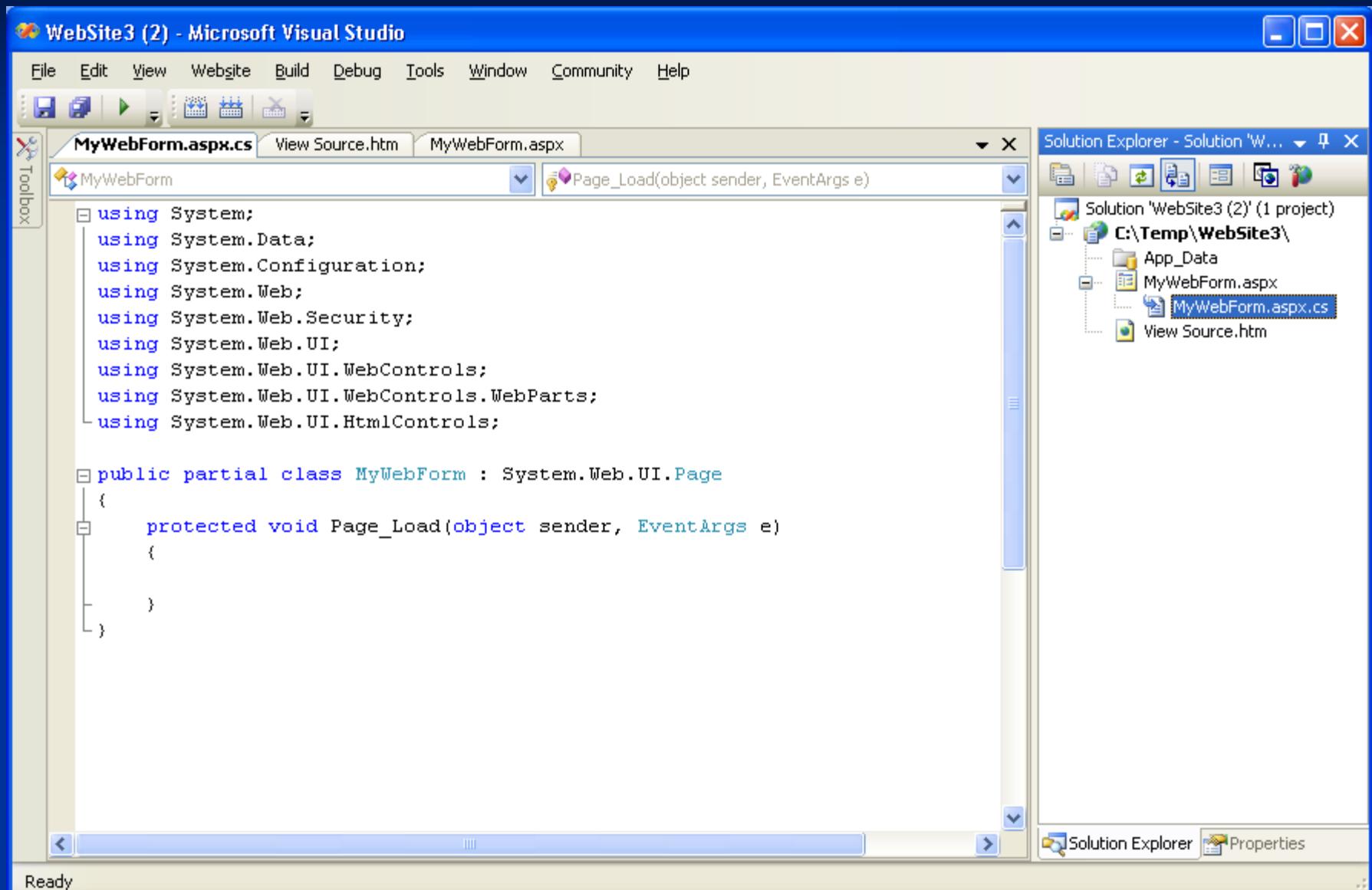
The screenshot shows a browser window with two tabs: "My Web Form" and "view-source:localhost:5948". The current view is "view-source:localhost:5948/MyWebForm.aspx". The page content is a raw HTML document with line numbers from 1 to 36 on the left. The HTML includes a DOCTYPE declaration, an XML namespace, a head section with a title and CSS style for a form, and a body section containing a form with various input fields and a submit button.

```
1
2
3 <!DOCTYPE html>
4
5 <html xmlns="http://www.w3.org/1999/xhtml">
6 <head><title>
7     My Web Form
8 </title>
9     <style type="text/css">
10    #form1 {
11        height: 200px;
12        width: 600px;
13    }
14    </style>
15 </head>
16 <body style="height: 190px; width: 681px">
17     <form method="post" action="MyWebForm.aspx" id="form1">
18         <div class="aspNetHidden">
19             <input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE"
20             value="12UFDvWqk+RzPbGQWXF1chGstDi7psbS+/OorDP7PH1wjB+vqxe2wwd2yKHQYgSAVSKDQZu4Ky+PQHV7Q4Y5aCktPieygg4
21             grxtGz7l+KeM=" />
22         </div>
23
24         <div class="aspNetHidden">
25
26             <input type="hidden" name="__EVENTVALIDATION" id="__EVENTVALIDATION"
27             value="1+Vg8Rk3bf7SQLldPrJ2Cam20GiBiqQb9wgA+cS6S2Z1P5W+Ow8pFc1GvwiLUaCCe3QnxS4Krn8Q6WW28qlQjxqkAMqj4U
28             th9WLtSt3StM8OzBVZfqKSnrVueticYxvDZ2CeEHGTfwPRUSQ9kIGY/UieyQLItLcM+Go70+uY=" />
29         </div>
30         Count:
31         <input name="Count" type="text" value="3" id="Count" />
32         <p></p>
33         <input type="submit" name="IncrementButton" value="Increment" id="IncrementButton" />
34         <p></p>
35         <input type="submit" name="Button1" value="Other" id="Button1" />
36
37     </form>
38 </body>
39 </html>
```

Rendered Web Form



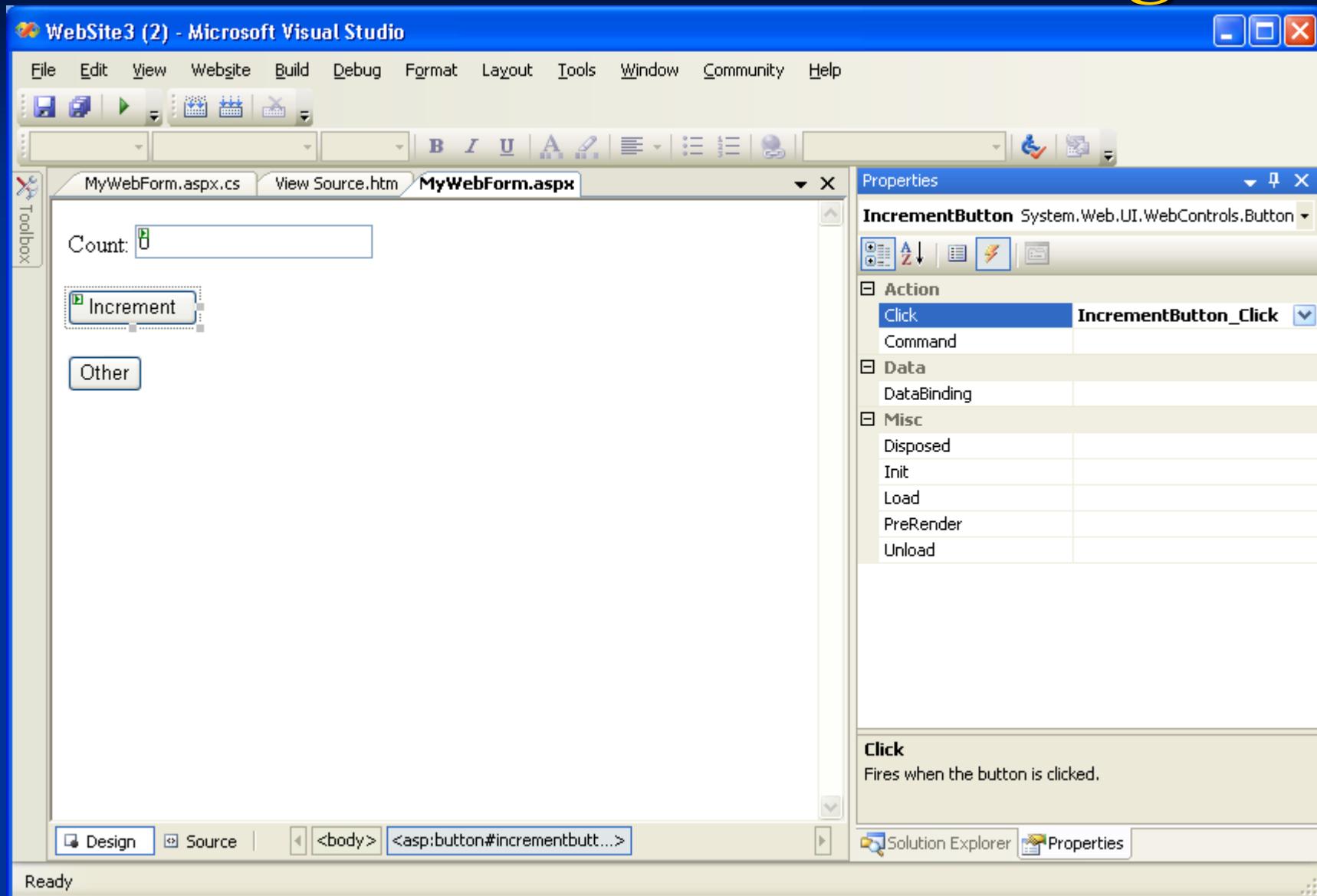
Viewing Code Behind Files



Web Forms Event Model

- Previously, client-side user input events could only be handled by client-side scripts (VBScript or ECMAScript).
- ASP.NET allows client-side events to be transparently dealt with by server-side code.
- Events such as button pushes cause the form to be immediately posted to the server together with an encoding of the event.
- Other events, such as text changed events are processed by the server in a batch fashion when the next post event occurs.
- Can still handle events on the client-side using traditional client-side scripting languages.

Web Form Event Handling



MyWebForm.aspx

The screenshot shows the Microsoft Visual Studio interface with the title bar "Diary - Microsoft Visual Studio". The menu bar includes FILE, EDIT, VIEW, WEBSITE, BUILD, DEBUG, TEAM, SQL, TOOLS, TEST, ARCHITECTURE, ANALYZE, WINDOW, and HELP. The toolbar has icons for file operations like Open, Save, and Print. The status bar at the bottom shows "Ready", "Ln 20", "Col 74", "Ch 74", and "INS". The code editor window displays the ASPX page source:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="MyWebForm.aspx.cs" Inherits="MyWebForm" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title> My Web Form </title>
    <style type="text/css">
        #form1 {
            height: 200px;
            width: 600px;
        }
    </style>
</head>
<body style="height: 190px; width: 681px">
    <form id="form1" runat="server">
        Count:
        <asp:TextBox ID="Count" runat="server"> 0 </asp:TextBox>
        <p></p>
        <asp:Button ID="IncrementButton" runat="server" Text="Increment" OnClick="IncrementButton_Click" />
        <p></p>
        <asp:Button ID="Button1" runat="server" Text="Other" />

    </form>
</body>
</html>
```

The "Source" tab is selected in the bottom navigation bar. The status bar indicates the current line is 20, column 74, character 74, and the mode is "INS".

Codebehind: MyWebForm.cs

The screenshot shows the Microsoft Visual Studio interface with the title bar "Diary - Microsoft Visual Studio". The menu bar includes FILE, EDIT, VIEW, WEBSITE, BUILD, DEBUG, TEAM, SQL, TOOLS, TEST, ARCHITECTURE, ANALYZE, WINDOW, and HELP. The toolbar has icons for file operations like Open, Save, and Print, along with debugging tools. The status bar at the bottom shows "100 %", "Ready", "Ln 14", "Col 9", "Ch 9", and "INS". The code editor window displays the file "MyWebForm.aspx.cs" with the following content:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class MyWebForm : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (! IsPostBack)
            Count.Text = "1";

    }

    protected void IncrementButton_Click(object sender, EventArgs e)
    {
        Count.Text = (Int32.Parse(Count.Text) + 1).ToString();
    }
}
```

Partial Classes and Compiling ASP.NET pages

<http://msdn2.microsoft.com/en-us/library/ms178138.aspx>

ASP.NET Web Site Pre-Compilation

<http://msdn2.microsoft.com/en-us/library/399f057w.aspx>

Web Form Lifecycle

Server Side

1. Create web page object
2. Construct object
3. Init event
4. *Restore view state*
5. *Process post data*
6. Page_Load event
if (!IsPostBack) ...
7. *Handle posted events*
8. Save view state
9. Render to HTML

Client Side

100% pure HTML
and JavaScript
(some auto-generated)

- Hidden view state
- Major events cause postback.

Web Form Lifecycle

Server Side Client Side

MyWebForm object

TextBox1:

```
Text:          "20"  
EventTextChanged: null  
ReadOnly:      false  
...
```

Button1:

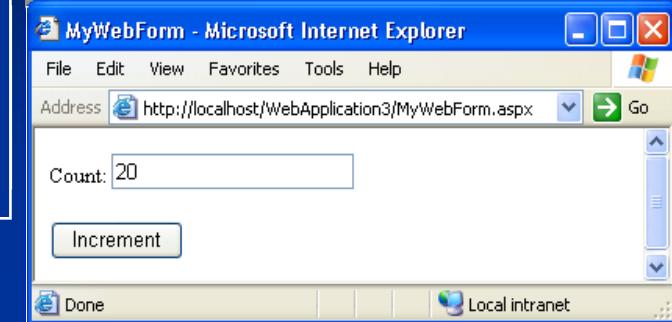
```
Text:          "Increment"  
EventClick:    Butt1_Click  
...
```

HTTP Request

```
MyWebForm.aspx  
?ViewState=dg3sg32  
(TextBox.Text="1")  
&TextBox1=19&  
Button1=Increment
```

HTTP Response

```
<html>  
<input ViewState ...  
value="jk143kls"/>  
(TextBox1.Text="20")  
<input TextBox1 ...  
value="20" />  
<input Button1 .../>  
</html>
```



Step 7. Process Posted Events
(IsPostBack = true)

Exotic Web Controls

WebSite3 (2) - Microsoft Visual Studio

File Edit View Website Build Debug Format Layout Tools Window Community Help

Toolbox

MyWebForm.aspx*

Solution Explorer - Solution 'We...' X

March 2007

Mon	Tue	Wed	Thu	Fri	Sat	Sun
26	27	28	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1
2	3	4	5	6	7	8

Log In

User Name: *

Password: *

Remember me next time.

Log In

Design Source <body> <asp:calendar#calendar1>

Solution Explorer Properties

Ready

The screenshot shows the Microsoft Visual Studio IDE interface for a web application named 'WebSite3 (2)'. The main window displays a web form titled 'MyWebForm.aspx*'. On the left, the 'Toolbox' is open, showing various web controls like CheckBoxList, RadioButton, RadioButtonList, Image, ImageMap, Table, BulletedList, HiddenField, Literal, and Calendar. The 'Calendar' control is currently selected. Below the toolbox, there are sections for Data, Validation, and Navigation. The central area contains a calendar for March 2007 and a 'Log In' form with fields for User Name and Password, a 'Remember me next time.' checkbox, and a 'Log In' button. The status bar at the bottom indicates 'Design' mode, the source code, and the current control under edit (<asp:calendar#calendar1>). To the right, the 'Solution Explorer' shows a single project with files: App_Data, MyWebForm.aspx, MyWebForm.aspx.cs, and Web.Config. The 'Properties' window is also visible. The title bar at the top says 'WebSite3 (2) - Microsoft Visual Studio'.

Exotic Web Controls

The screenshot shows the Microsoft Visual Studio interface for a web application named "WebSite3 (2)". The main window displays the source code for a file named "MyWebForm.aspx". The code includes the following structure:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="MyWebF.aspx.cs" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>My Web Form </title>
</head>
<body>
    <form id="form1" runat="server">
        <asp:Calendar ID="Calendar1" runat="server">
        </asp:Calendar>
        <asp:Login ID="Login1" runat="server">
        </asp:Login>
    </form>
</body>
</html>
```

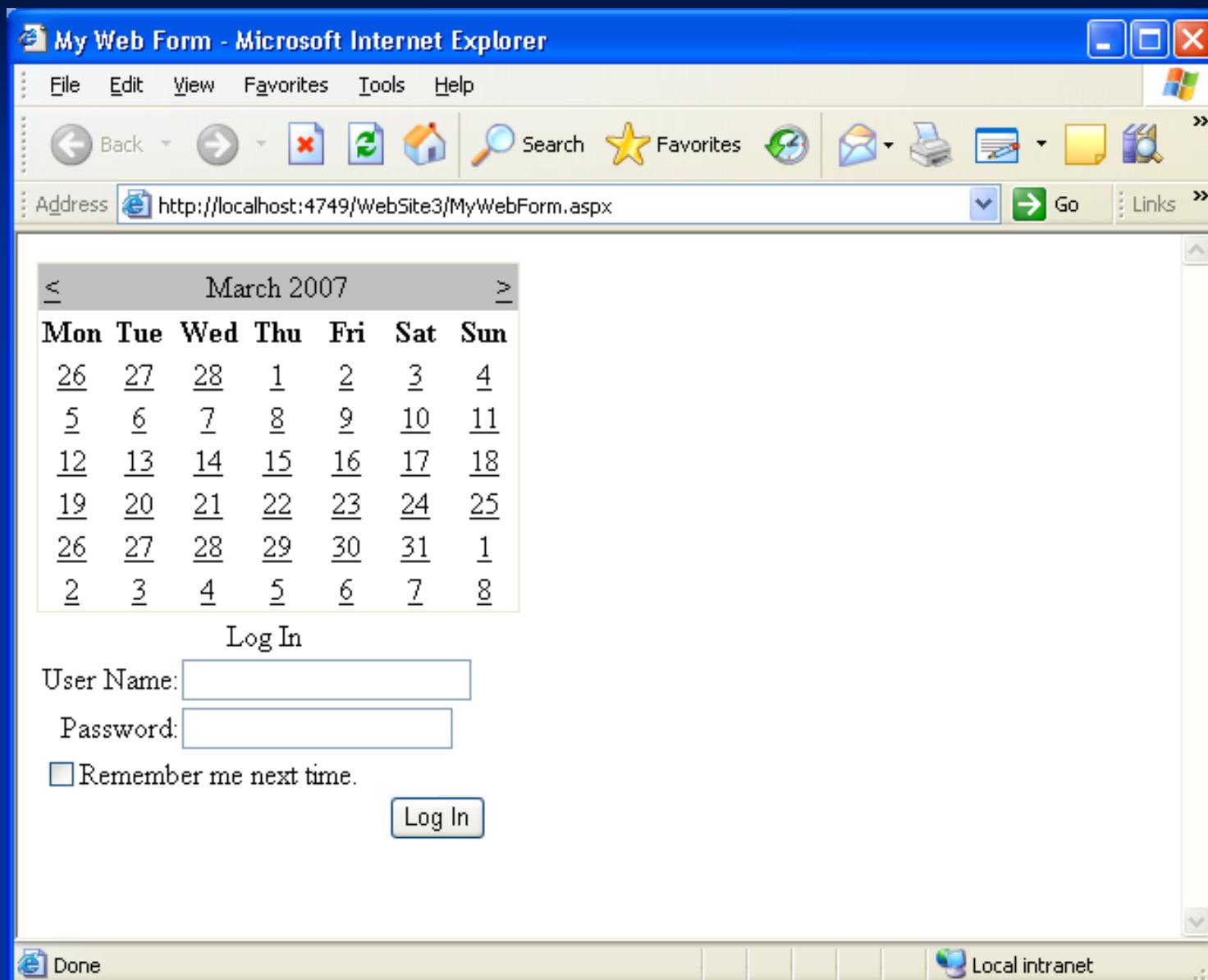
The "Toolbox" on the left lists various ASP.NET controls, with the "Calendar" control currently selected. The "Solution Explorer" on the right shows the project structure with files like "MyWebForm.aspx", "MyWebForm.aspx.cs", and "Web.Config". The status bar at the bottom provides information about the current line (Ln 6), column (Col 32), character (Ch 32), and mode (INS).

Exotic Web Controls Render as HTML

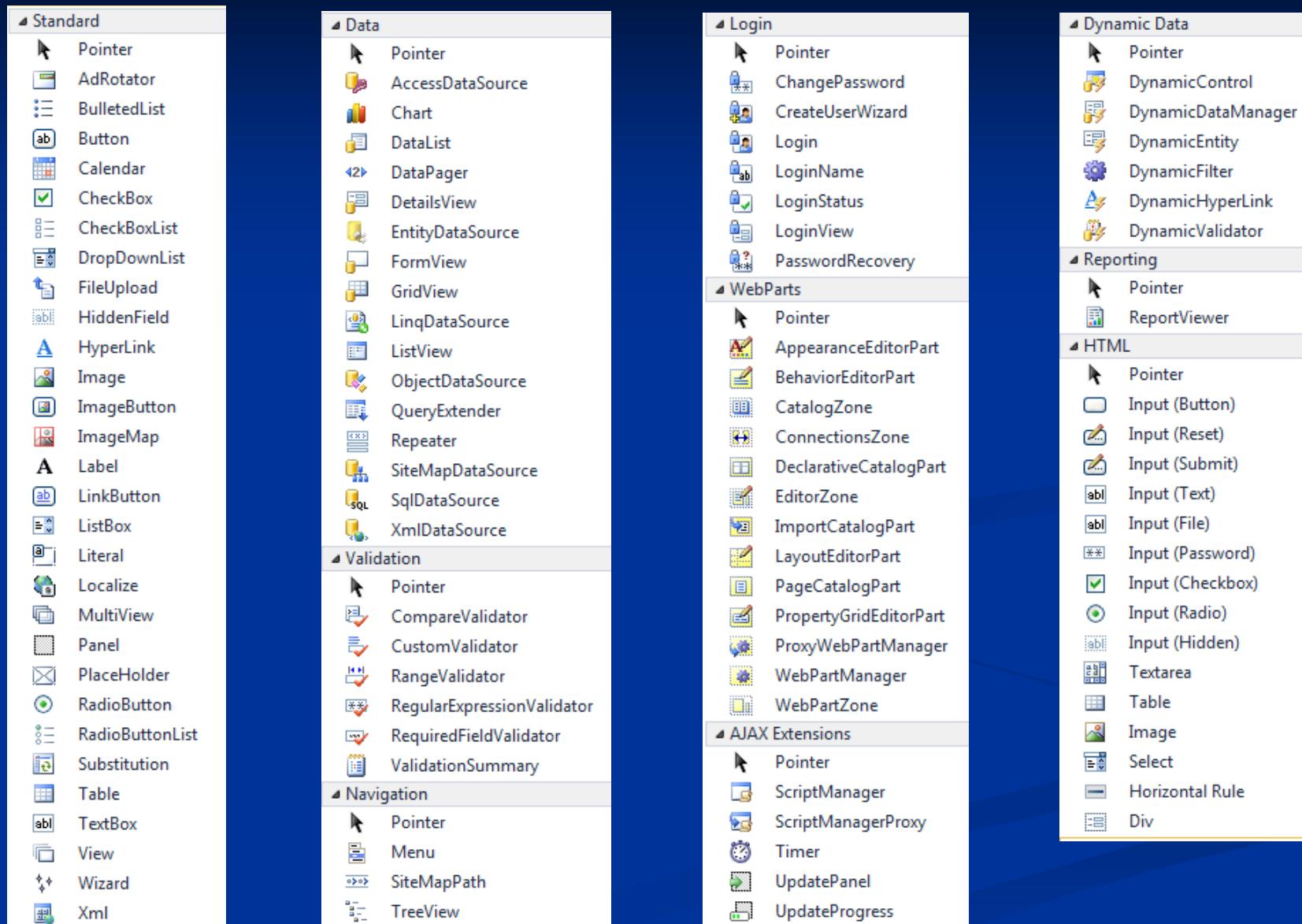
The screenshot shows a web development environment with the following details:

- Code View:** The main pane displays the HTML source code for a web page.
- Structure View:** A tree view on the left shows the document structure:
 - <html>
 - <head>
 - <title>My Web Form </title>
 - </head>
 - <body>
 - <form name="form1" method="post" action="MyWebForm.aspx" onsubmit="javascript:return WebForm_OnSubmit();"
 - id="form1">
 - <div>...</div>
 - <script>...</script>
 - <script src="/WebSite3/WebResource.axd?d=xb5Sqa2qVM2zhXRnpvfKrQ2&t=632969105401853548" type="text/javascript"></script>
 - <script src="/WebSite3/WebResource.axd?d=3TfefCACbiOgx_BzAx1-P1Ini3iBGBYaeCK_ZAnOpus1&t=6329691054018535" type="text/javascript"></script>
 - <script>...</script>
 - <table id="Calendar1" cellspacing="0" cellpadding="2" title="Calendar" border="0" style="border-width: 1px; border-style: solid; border-collapse: collapse;">
 - <tr>
 - <td colspan="7" style="background-color: Silver;">
 - <table cellspacing="0" border="0" style="width: 100%; border-collapse: collapse;">
 - <tr>
 - <td style="width: 15%;">
 - <a href="javascript:_doPostBack('Calendar1','V2588')" style="color: Black" title="G <"></td>
 - <td align="center" style="width: 70%;">
 - March 2007</td>
- Status Bar:** At the bottom, it shows tabs for "Design" and "Source".
- Page Number:** In the bottom right corner, it says "Week 5 – Slide 24".

Exotic Web Controls Render as HTML



Web Controls



Data Binding

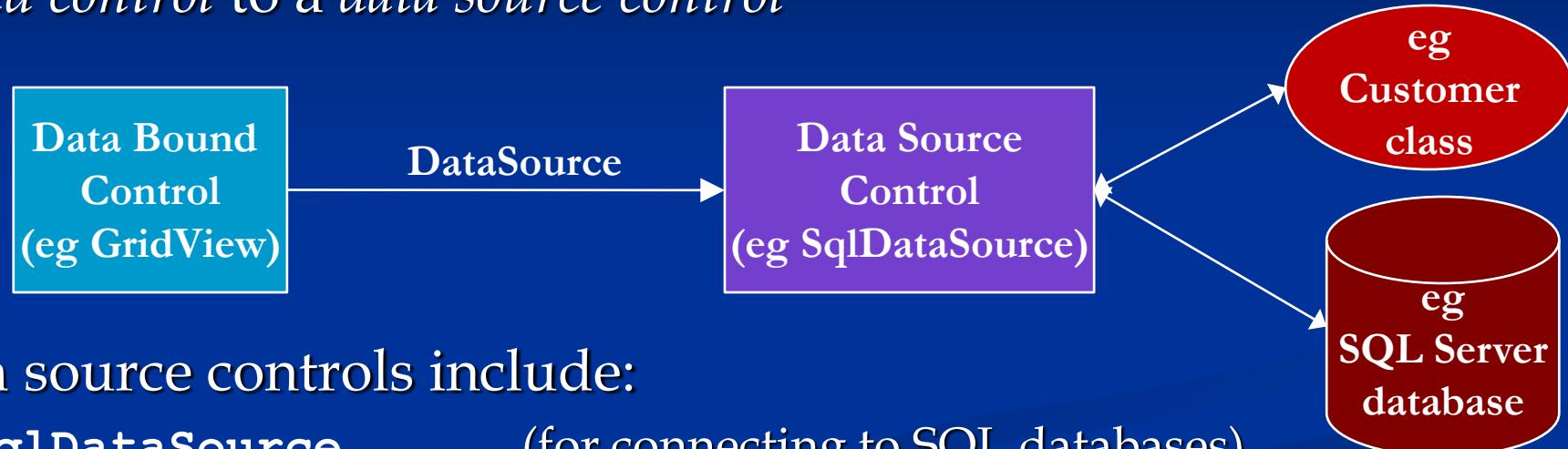
- Many simple web controls such as Buttons, Labels and Textboxes have a property called Text that you can get and set.

```
string content = this.TextBox1.Text;  
this.Button2.Text = "Press Me";
```

- Other “*container*” web controls such as GridView, FormView, DetailsView, DropDownList and Repeater are designed to render a list or set of values.
- These controls can be initialized by *binding* them to a *data source* that represents a collection of values
 - this can be done either *declaratively* or *programmatically*.

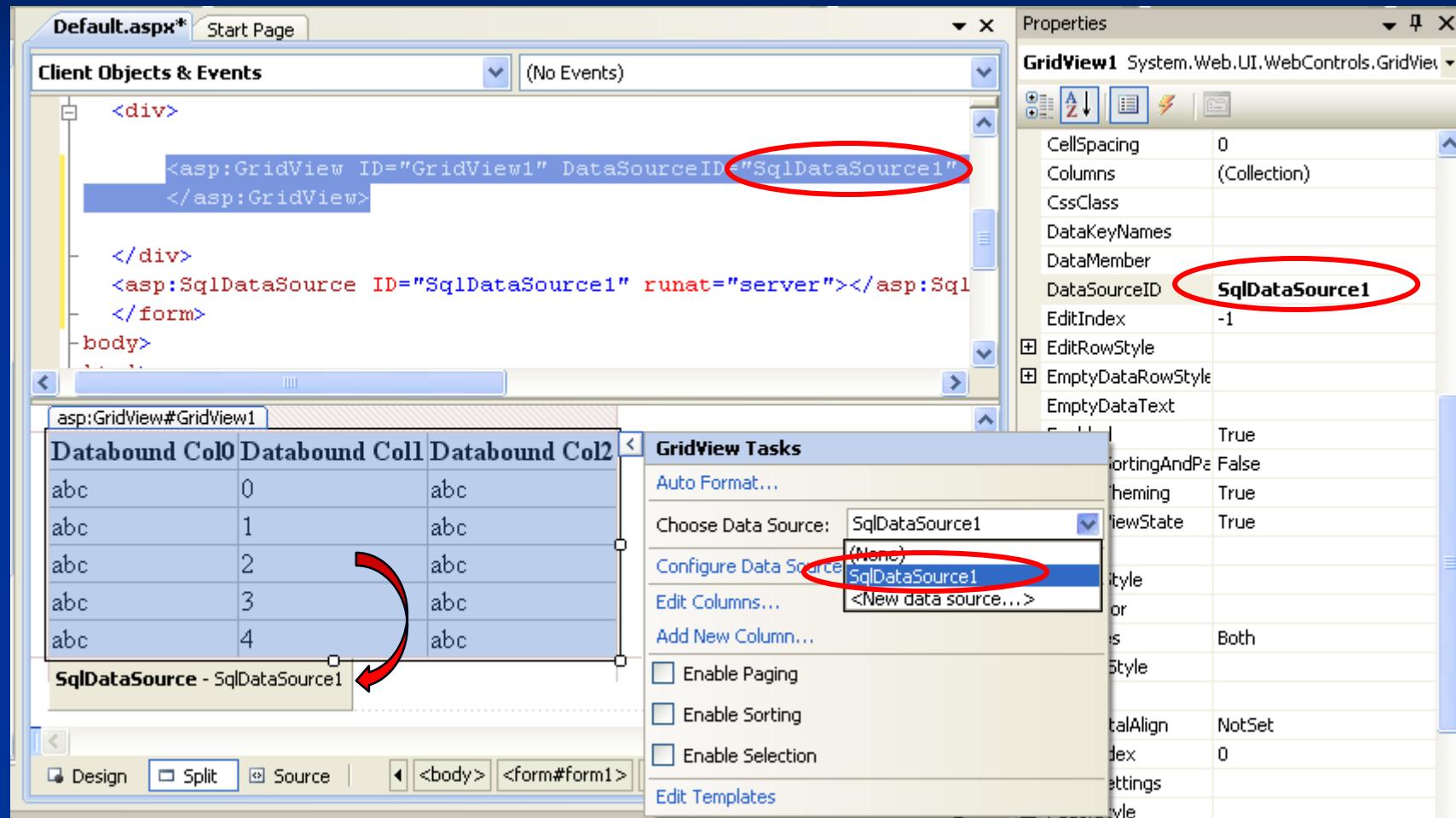
Declarative Data Binding

- Data Binding can be done in a *declarative* way by binding a *data bound control* to a *data source control*



- Data source controls include:
 - **SqlDataSource** (for connecting to SQL databases)
 - **AccessDataSource** (for connecting to MS access databases)
 - **ObjectDataSource** (**for connecting to business objects**)
 - **XmlDataSource** (for connecting to XML documents)
 - **SiteMapDataSource** (for connecting to an XML site map)
- Data source controls are added and configured like other web controls but aren't rendered on the client machine.

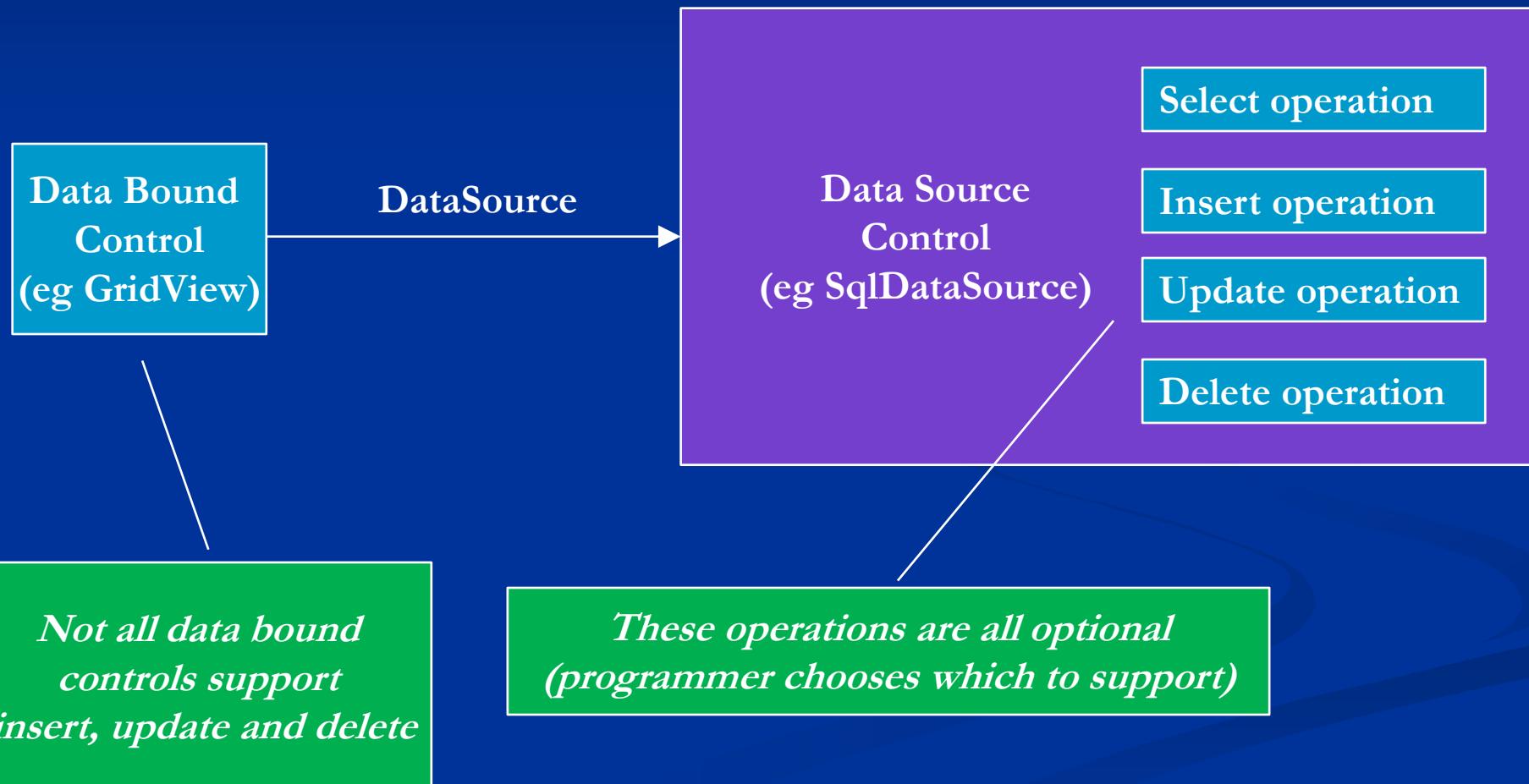
Configuring DataSourceID



Data Source Controls

- Data Source Controls need to be configured to deal with Select, Insert, Update and Delete operations
 - in the case of SqlDataSource and AccessDataSource, these operations are expressed as *SQL commands* executed on a specific database.
 - in the case of ObjectDataSource, these operations are expressed as .NET code, typically contained in the Business logic layer.
- When a data bound control is first rendered, it *asks* its data source control for an initial set of records to display (*Select*).
- If the client's interaction with the data bound control results in a record being *removed*, then when this event is propagated back to the server, the data bound control *asks* its data source control to perform a *Delete* operation.
 - similarly for update and insert operations.

Data Source Operations



Programmatic Data Binding

- Prior to data source controls in ASP.NET 2.0, data binding had to be done programmatically.
 - Declarative data binding is generally preferable, but programmatic data binding is more flexible and necessary in some advanced cases.

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        this.GridView1.DataSource = myCollection;
        this.GridView1.DataBind();
    }
}
protected void GridView1_RowDeleted(object sender, GridViewDeletedEventArgs e)
{
    myCollection.Remove(e.Keys[0]);
}
```

Event handler for
RowDeleted event
on GridView1

Any collection type,
eg array, List<>, Dataset

Do the actual
data binding now!

Mapping Data Bound Fields

- Data Sources (regardless of kind) provide data in a form equivalent to what would be found in a relational database table, i.e. in general the data will consist of multiple rows or records, and each record will consist of one or more fields or columns.
- Some data bound controls (such as GridView and DataList) allow multiple records to be displayed simultaneously, others (such as DetailsView and FormView) only display one record at a time, but provide a means of scrolling.
- Each data bound control needs to be configured to specify if and how each field should be displayed.
 - for example a field might be mapped to a column on a GridView or to a nested web control in a FormView

Example: Mapping GridView Fields

The screenshot shows the Visual Studio IDE with the 'Default.aspx*' page open. On the left, the code editor displays the ASPX page with a GridView control and its columns defined.

```
<asp:GridView ID="GridView1" DataSourceID="SqlDataSource1" AutoGenerateColumns="False">
    <Columns>
        <asp:BoundField DataField="Product Name" SortExpression="Product Name" />
        <asp:BoundField DataField="English Name" SortExpression="English Name" />
        <asp:BoundField DataField="Quantity Per Unit" SortExpression="Quantity Per Unit" />
        <asp:BoundField DataField="Unit Price" SortExpression="Unit Price" />
        <asp:BoundField DataField="Units In Stock" SortExpression="Units In Stock" />
    </Columns>

```

The 'Fields' dialog box is open on the right, showing the mapping configuration for the 'Product Name' column:

- Available fields:** Shows a tree view with 'BoundField' selected, and under it, 'Product Name', 'English Name', 'Quantity Per Unit', 'Unit Price', and 'Units In Stock'.
- Selected fields:** Shows the 'Product Name', 'English Name', 'Quantity Per Unit', 'Unit Price', and 'Units In Stock' fields listed.
- BoundField properties:** A table showing properties for the 'Product Name' field:

HtmlEncodeFormatS	True
InsertVisible	True
NullDisplayText	
ReadOnly	False
ShowHeader	True
SortExpression	Product Name
Visible	True
- Data:** Shows 'DataField' set to 'Product Name'.
- Styles:** Shows 'DataFormatString'.
- HeaderText:** Shows the placeholder 'The text within the header of this field.'
- Buttons:** Includes 'OK' and 'Cancel' buttons, and links for 'Convert this field into a TemplateField', 'Configure Data Source...', 'Refresh Schema', 'Edit Columns...', 'Add New Column...', 'Enable Paging', 'Enable Sorting', 'Enable Selection', and 'Edit Templates'.

Example: Mapping FormView fields

Default.aspx* Start Page

Client Objects & Events (No Events)

```
<asp:FormView ID="FormView1" runat="server" DataKeyNames="Product ID">
    <DataSourceID="SqlDataSource1">
    <EditItemTemplate>
        Product ID:
        <asp:Label ID="Product_IDLabel1" runat="server"
            Text='<%# Eval("[Product ID]") %>' />
        <br />
        Supplier ID:
        <asp:TextBox ID="Supplier_IDTextBox" runat="server"
            Text='<%# Bind("[Supplier ID]") %>' />
        <br />
        Category ID:
```

Supplier_IDTextBox DataBindings

Select the property to bind to. You can then bind it by selecting a field. Alternatively, you can bind it using a custom code expression.

Bindable properties:

- Enabled
- ReadOnly
- Text
- Visible

Show all properties

Binding for Text

Field binding:

Bound to: Supplier ID

Format: (None)

Sample:

Two-way databinding

Custom binding:

Code expression: Bind("[Supplier ID]")

OK Cancel Refresh Schema

FormView Tasks

- Auto Format...
- Choose Data Source: SqlData
- Configure Data Source...
- Refresh Schema
- Enable Paging
- Edit Templates

Design Split Source <asp:FormView#FormView1> <EditItemTemplate>

The screenshot shows the Visual Studio IDE interface. A 'FormView Tasks' dialog is open over a 'Supplier_IDTextBox' control in the 'EditItemTemplate' of a 'FormView1'. The 'Supplier_IDTextBox' is highlighted with a yellow selection bar. The 'Supplier_IDTextBox DataBindings' dialog is displayed, showing the 'Text' property is bound to the '[Supplier ID]' field. The 'Supplier_IDTextBox' control is also selected in the 'EditItemTemplate' of the 'FormView1'. The background shows the ASPX page with the FormView and its associated code.

Creating Web Controls

- Application programmers can create their own web controls that can be reused on multiple web forms or in multiple web applications.
- There are two basic techniques for creating web controls:
 - Web User Controls (.ascx):
 - Created in the same way as Web Forms - by composing a collection of standard web controls.
 - Consist of an ascx file which describes the GUI and a code behind file.
 - Custom ASP.NET Server Controls:
 - Developed programmatically.
 - More flexible than user controls - not limited to composing existing web controls.
 - Can completely determine appearance and behaviour of control.

References

- To learn more about ASP.NET, visit <http://asp.net>.