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.

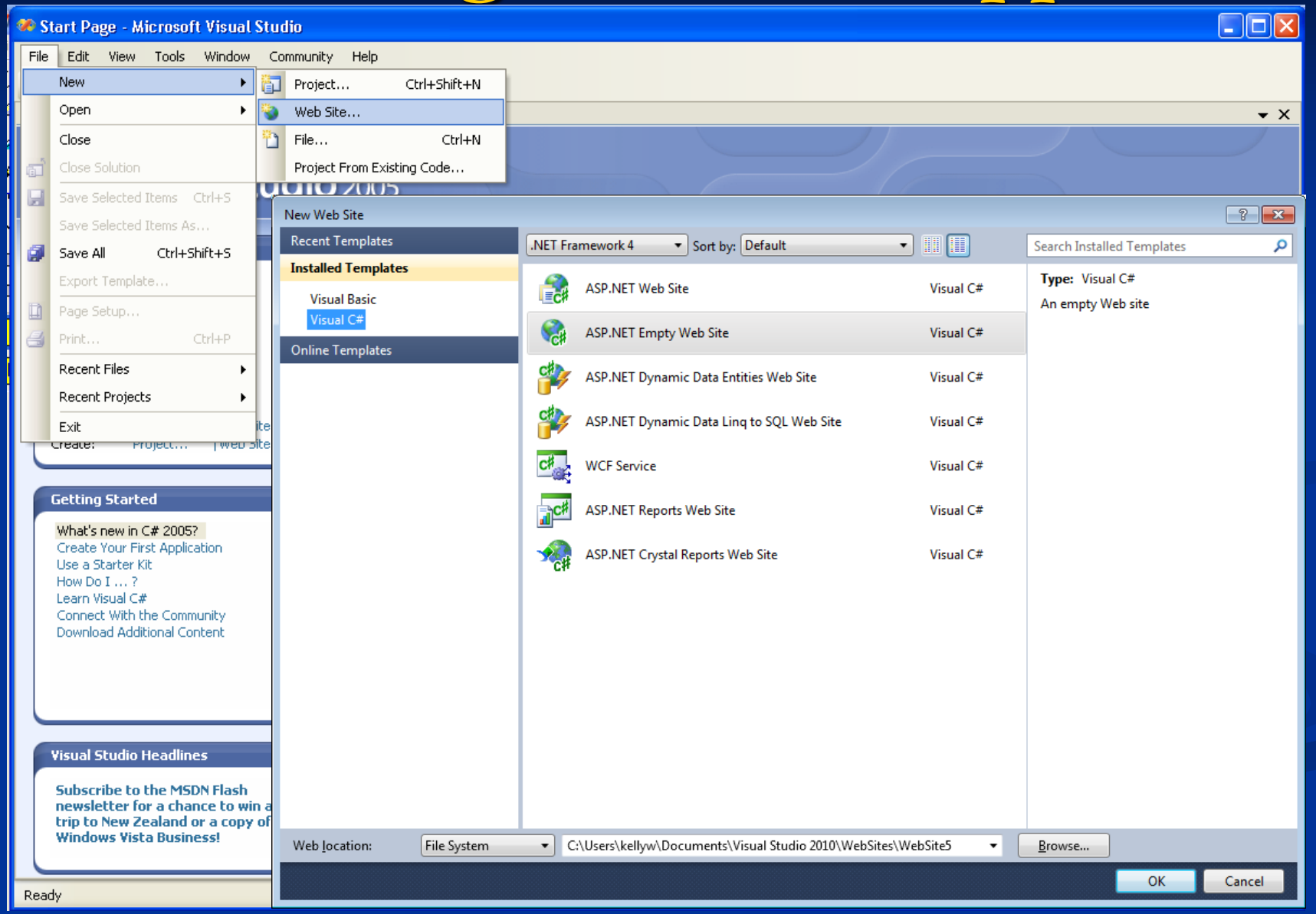
Traditional ASP Example

```
<%@ 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>
```

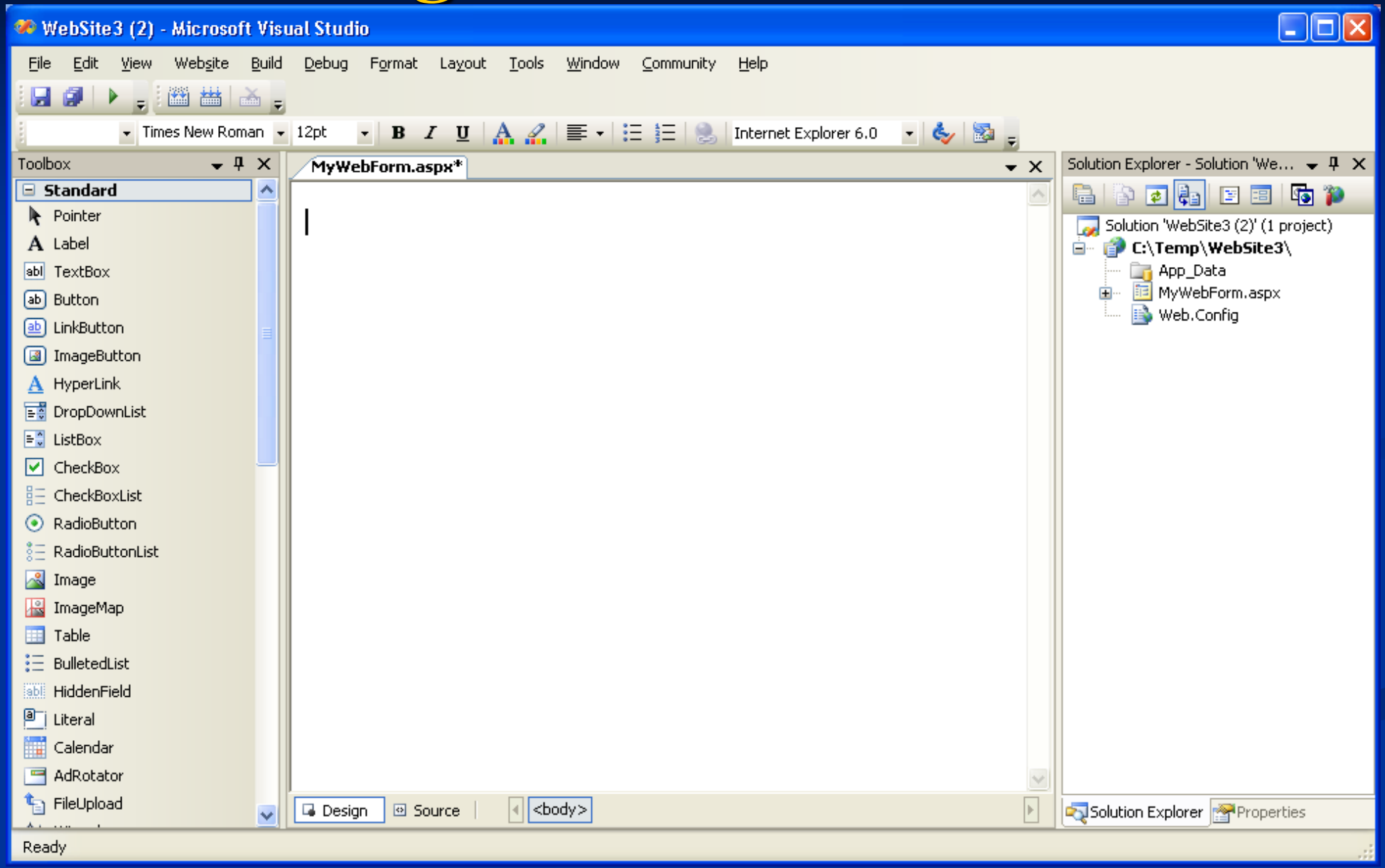
ASP.NET Features Overview

- 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

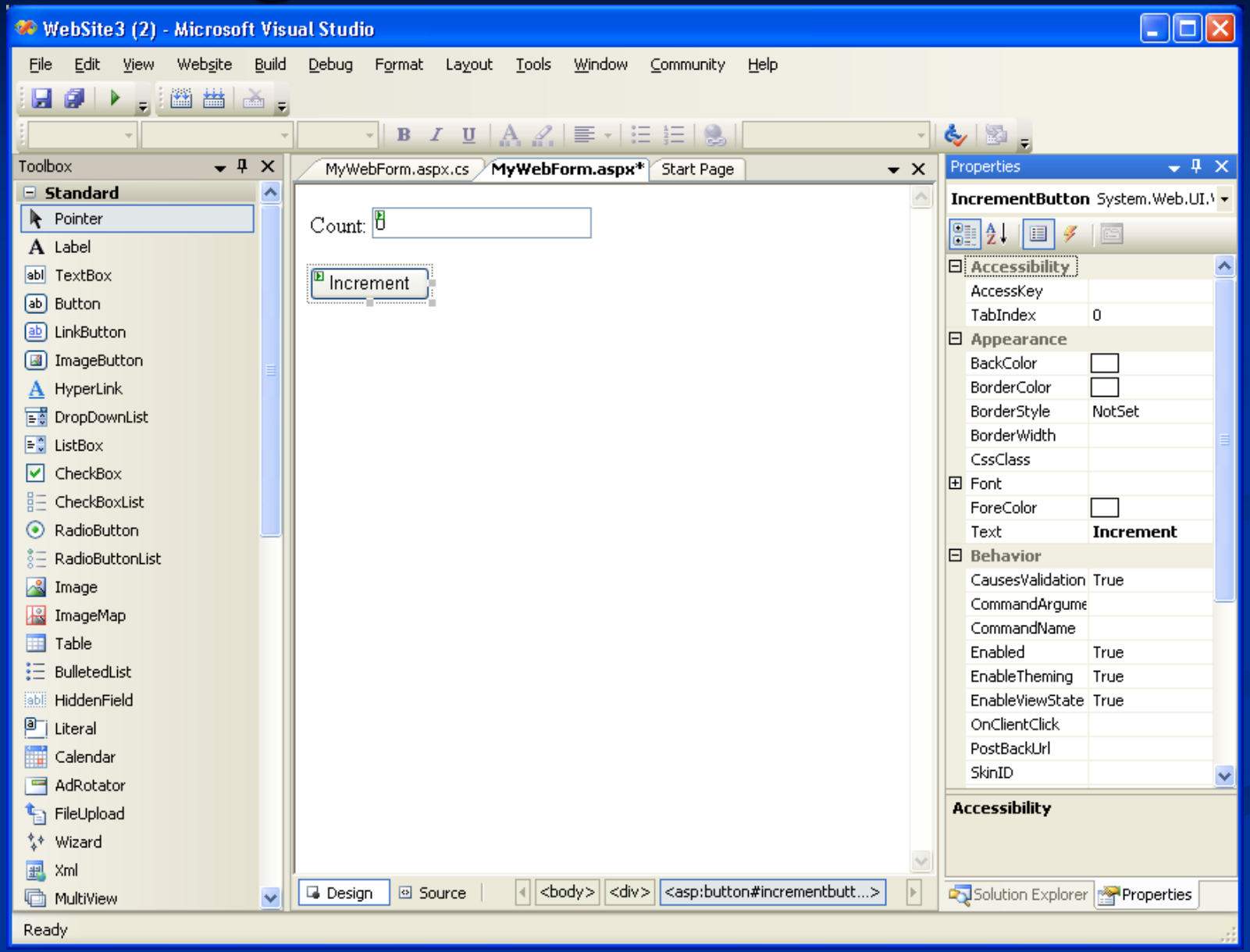
Creating ASP.NET Applications



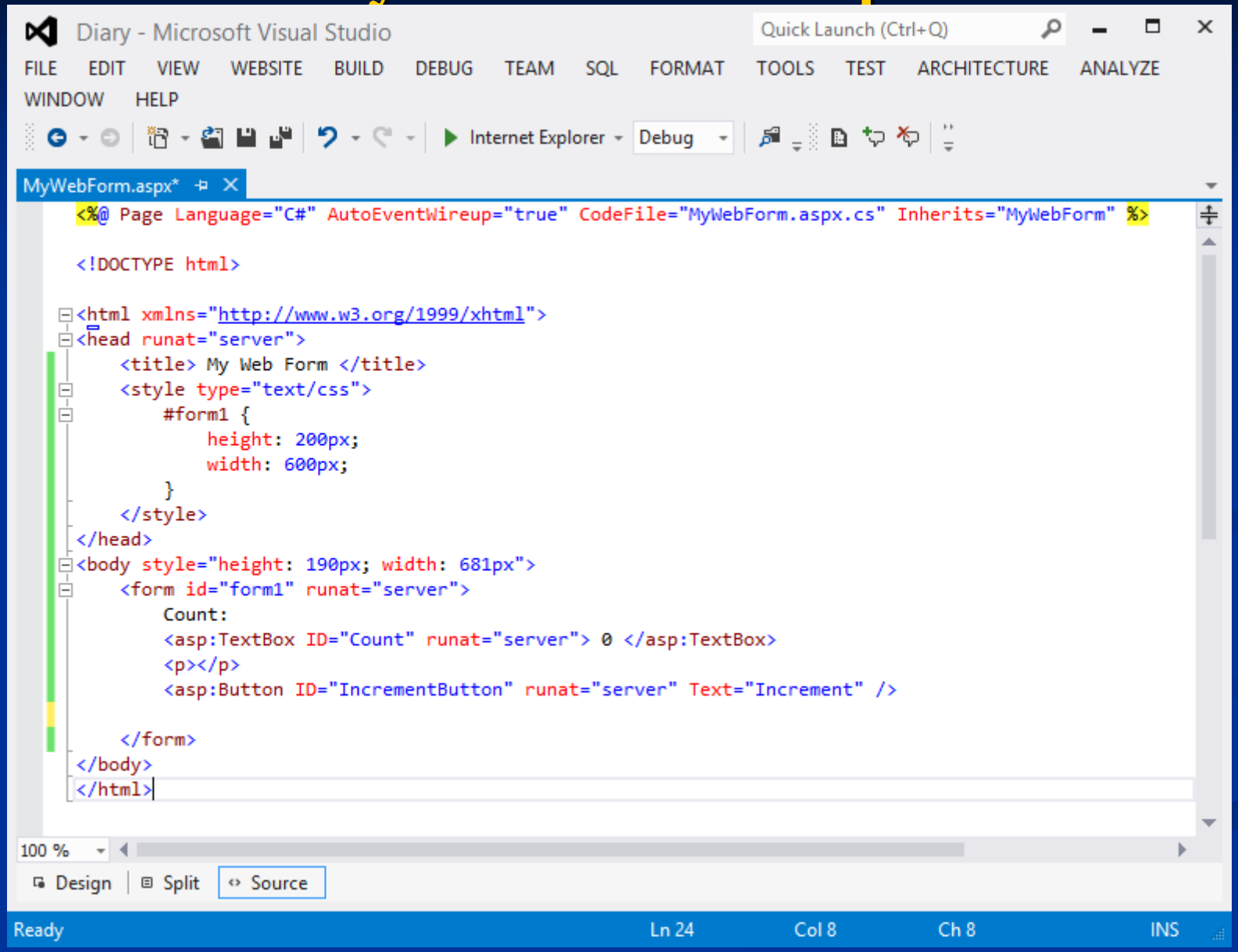
Visual Design of ASP.NET Web Forms



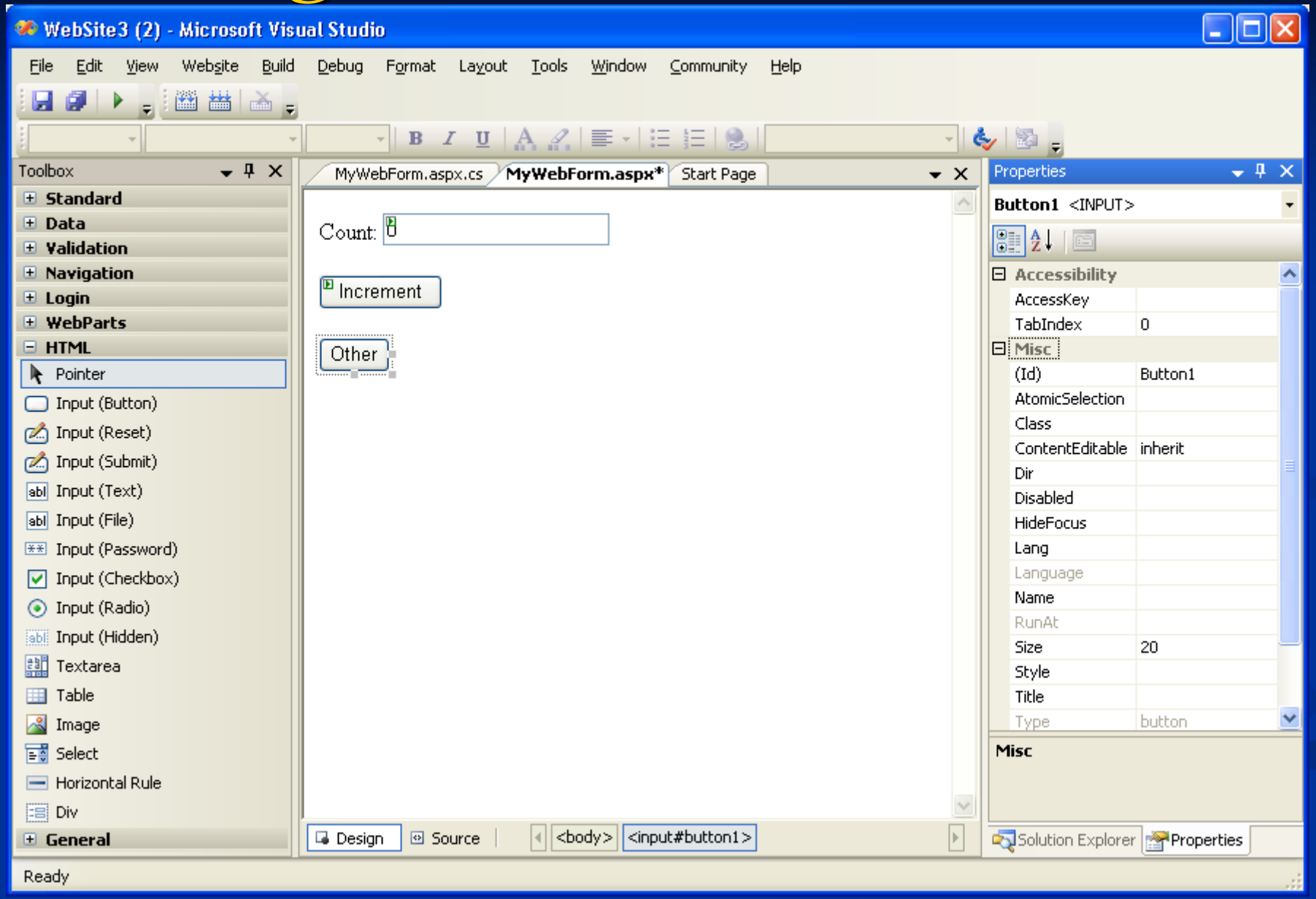
Adding ASP.NET Web Controls



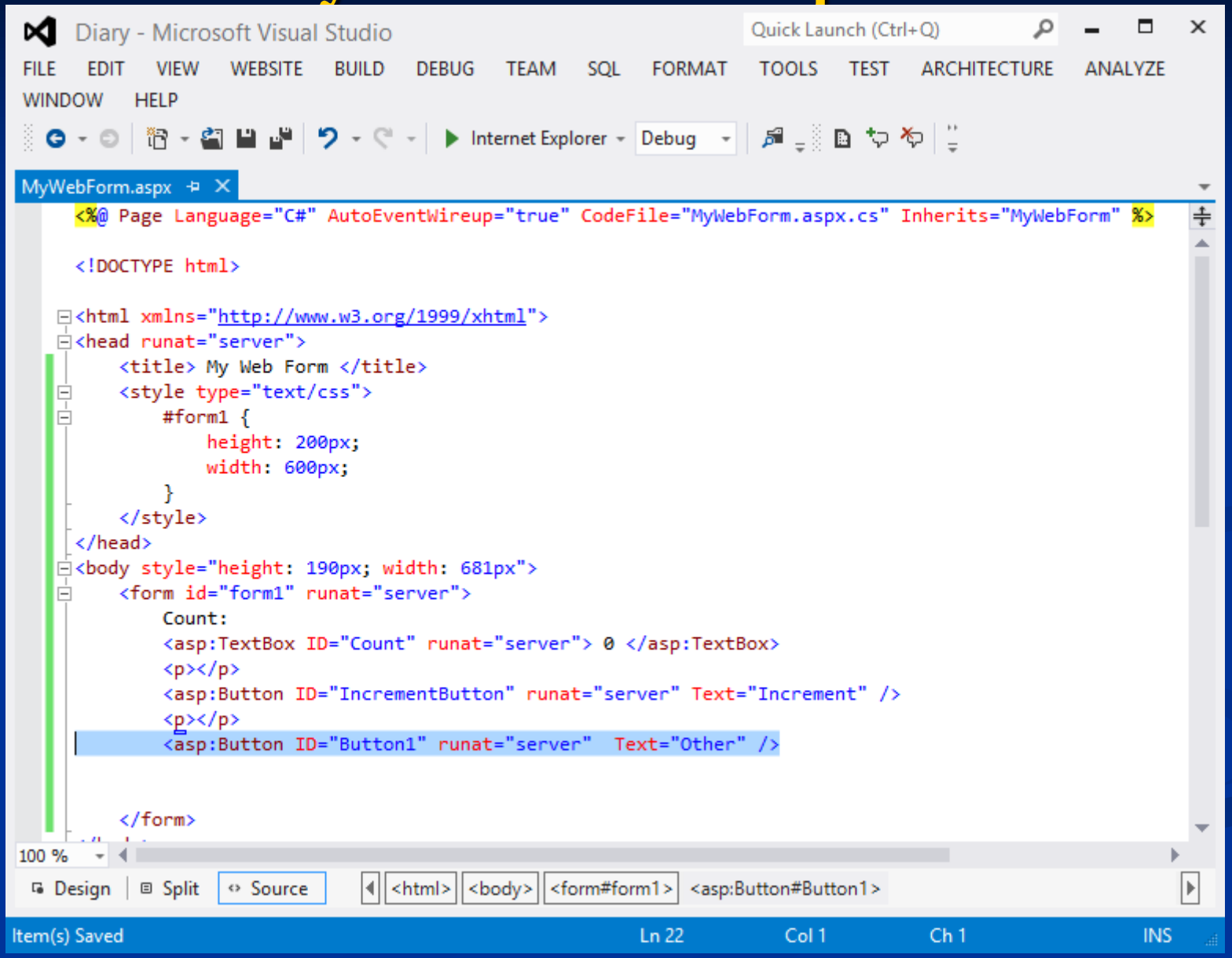
MyWebForm.aspx



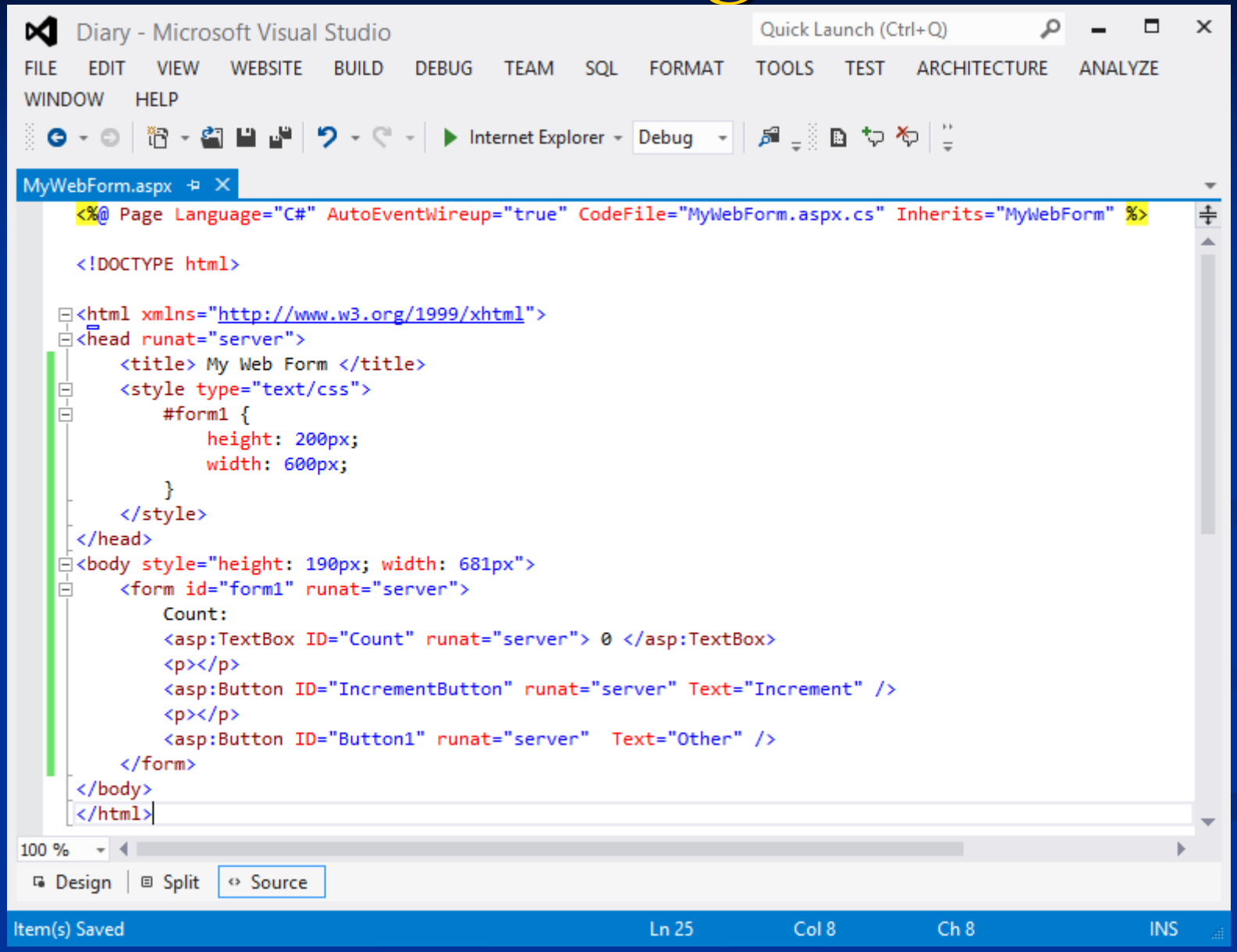
Adding Standard HTML Controls



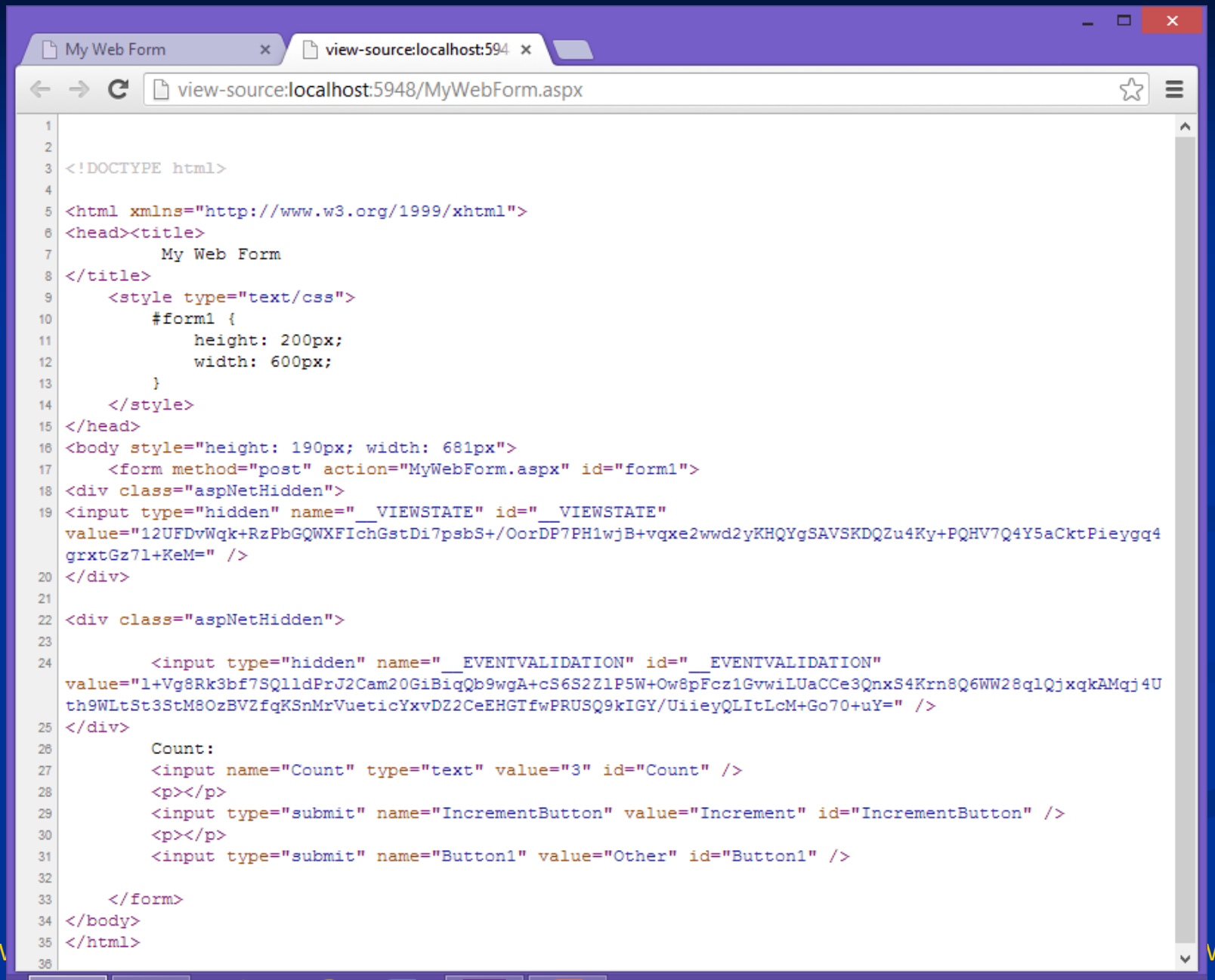
MyWebForm.aspx



Web Form containing ASP.NET tags

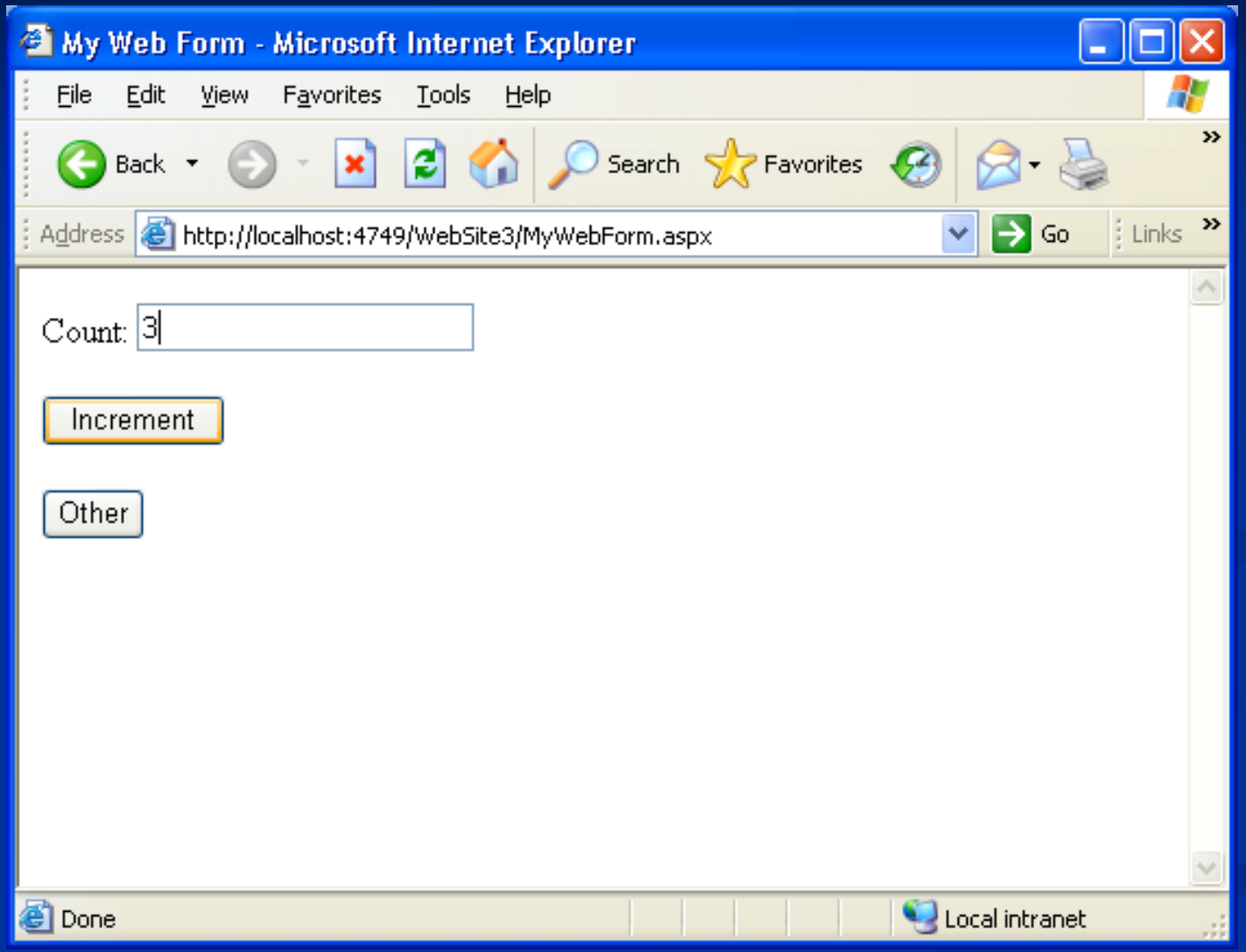


Pure HTML sent to Client

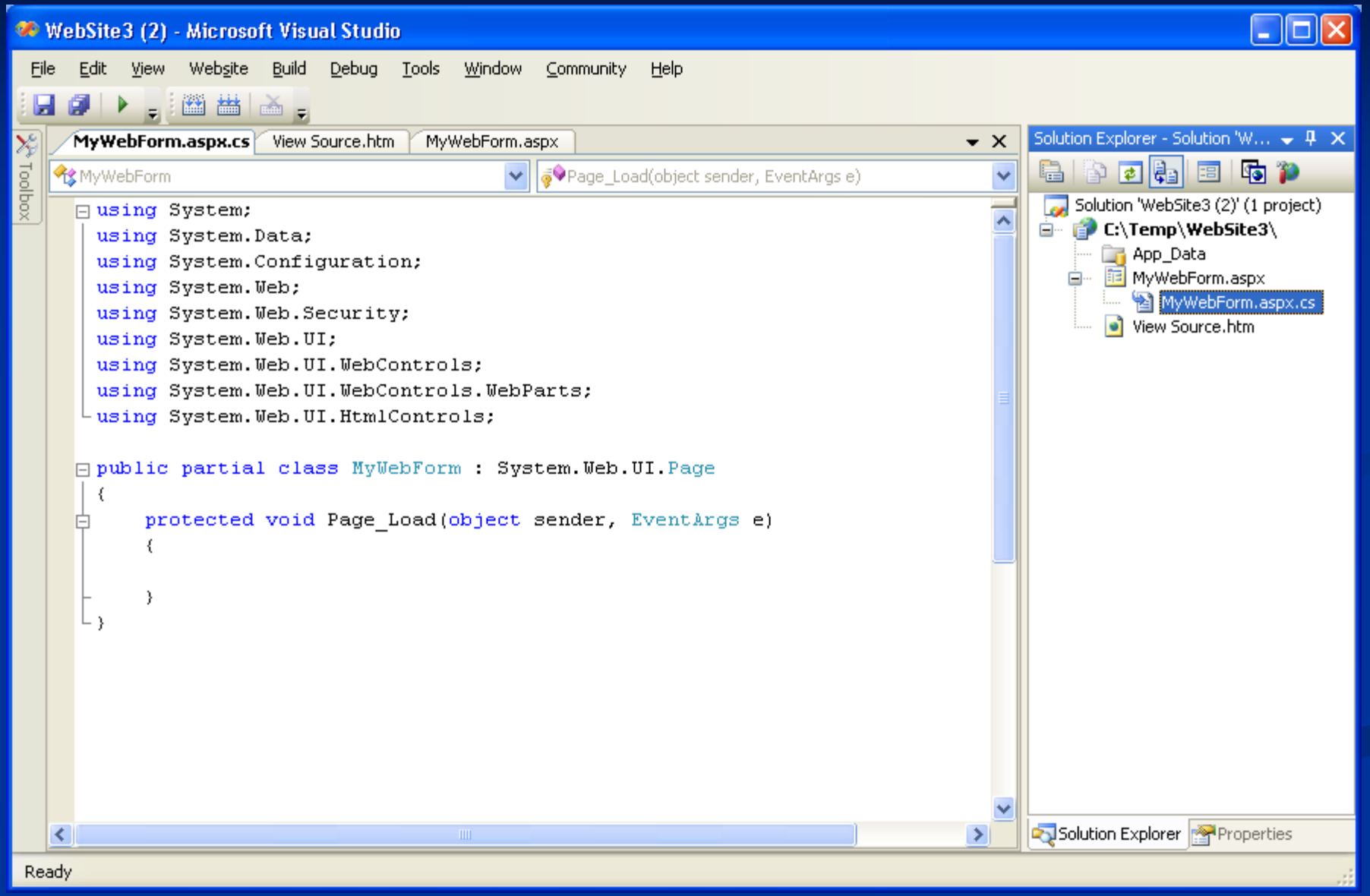



```
<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>
         My Web Form
</title>
    <style type="text/css">
        #form1 {
            height: 200px;
            width: 600px;
        }
    </style>
</head>
<body style="height: 190px; width: 681px">
    <form method="post" action="MyWebForm.aspx" id="form1">
<div class="aspNetHidden">
<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE"
value="12UFDvWqk+RzPbGQWXFIchGstDi7psbS+/OorDP7PH1wjB+vqxe2wwd2yKHQYgSAVSKDQZu4Ky+PQHV7Q4Y5aCktPieygq4
grxtGz7l+KeM=" />
</div>

<div class="aspNetHidden">

        <input type="hidden" name="__EVENTVALIDATION" id="__EVENTVALIDATION"
value="l+Vg8Rk3bf7SQlldPrJ2Cam20GiBiqQb9wgA+cS6S2ZlP5W+Ow8pFcz1GvwiLUaCCe3QnxS4Krn8Q6WW28qlQjxqkAMqj4U
th9WLtSt3StM8OzBVZfqKSnMrVueticYxvDZ2CeEHGTfwPRUSQ9kIGY/UiieyQLItLcM+Go70+uY=" />
</div>
        Count:
        <input name="Count" type="text" value="3" id="Count" />
        <p></p>
        <input type="submit" name="IncrementButton" value="Increment" id="IncrementButton" />
        <p></p>
        <input type="submit" name="Button1" value="Other" id="Button1" />

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

Rendered Web Form



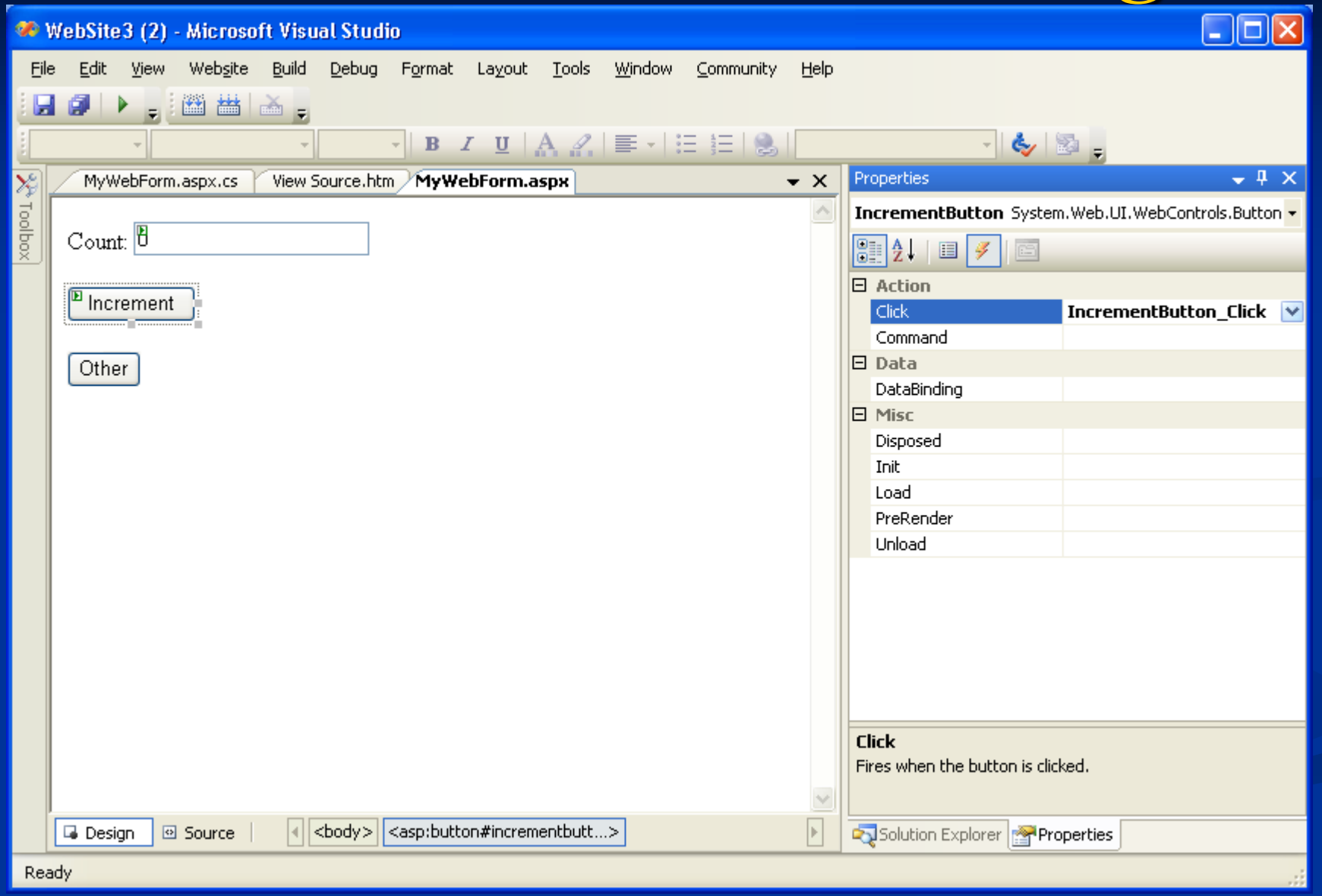
Viewing Code Behind Files



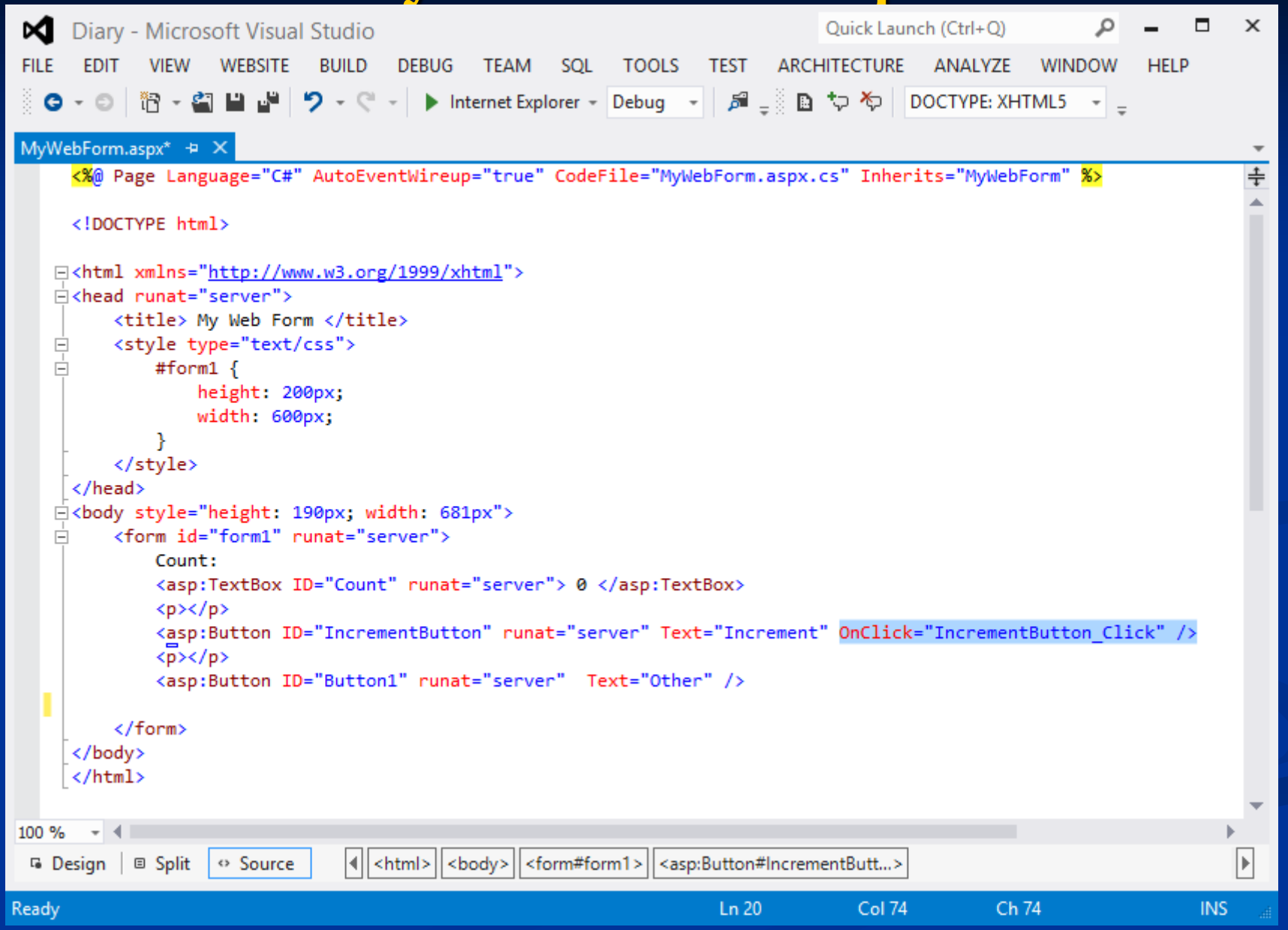
Web Forms Event Model

- 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 a 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.

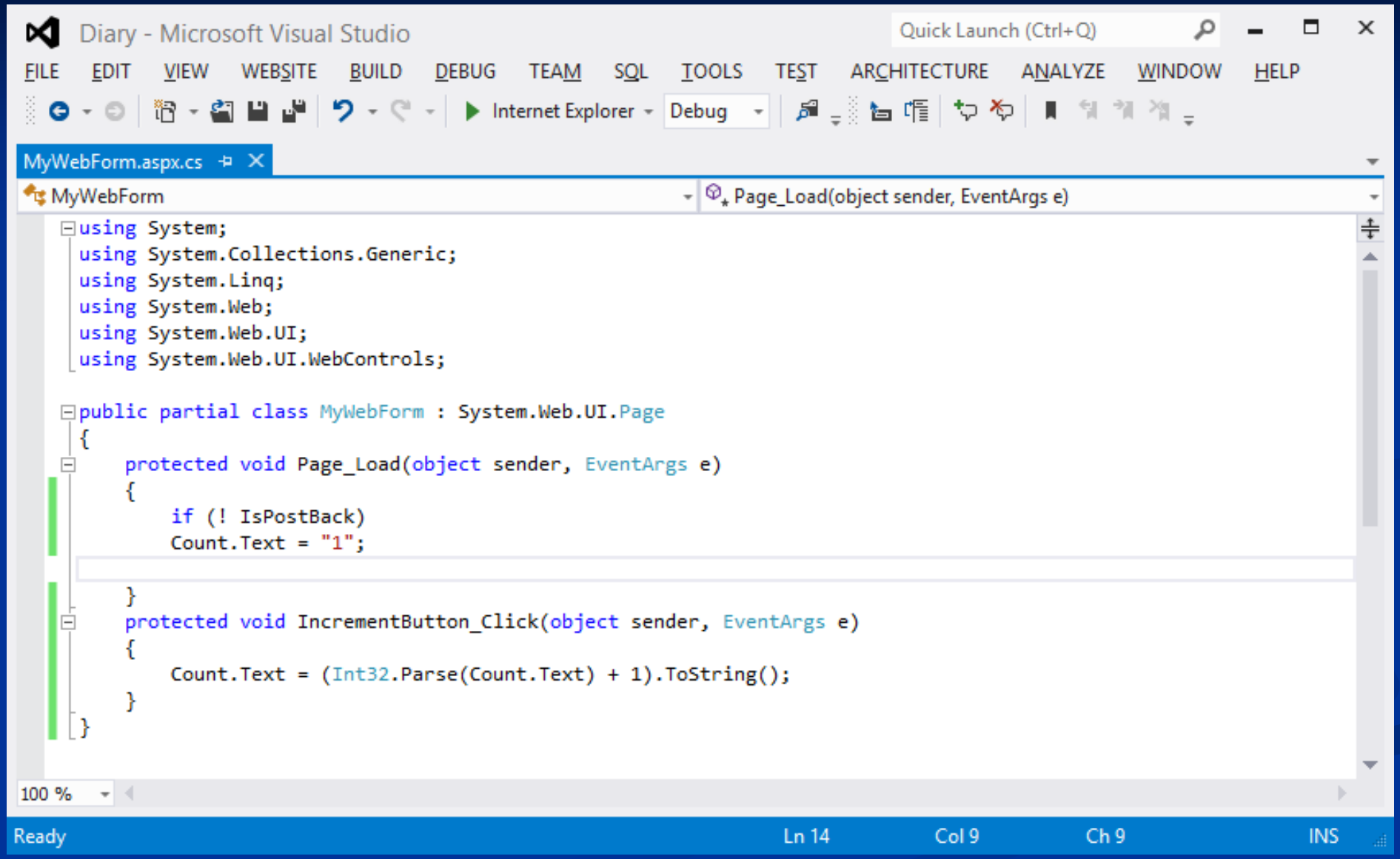
Web Form Event Handling



MyWebForm.aspx



Codebehind: MyWebForm.cs



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

response

request

Client Side

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

- Hidden view state

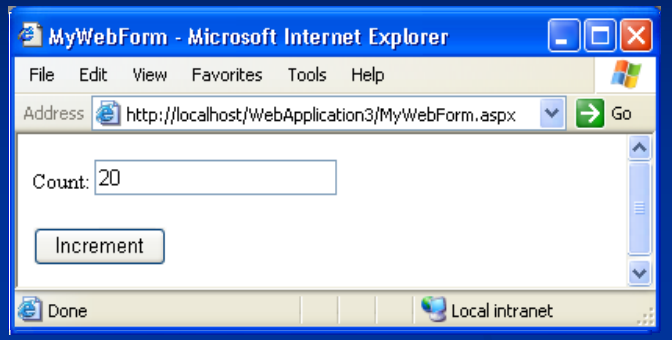
- *Major* events cause postback.

Web Form Lifecycle

Server Side

Client Side

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



MyWebForm object

TextBox1:

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

Button1:

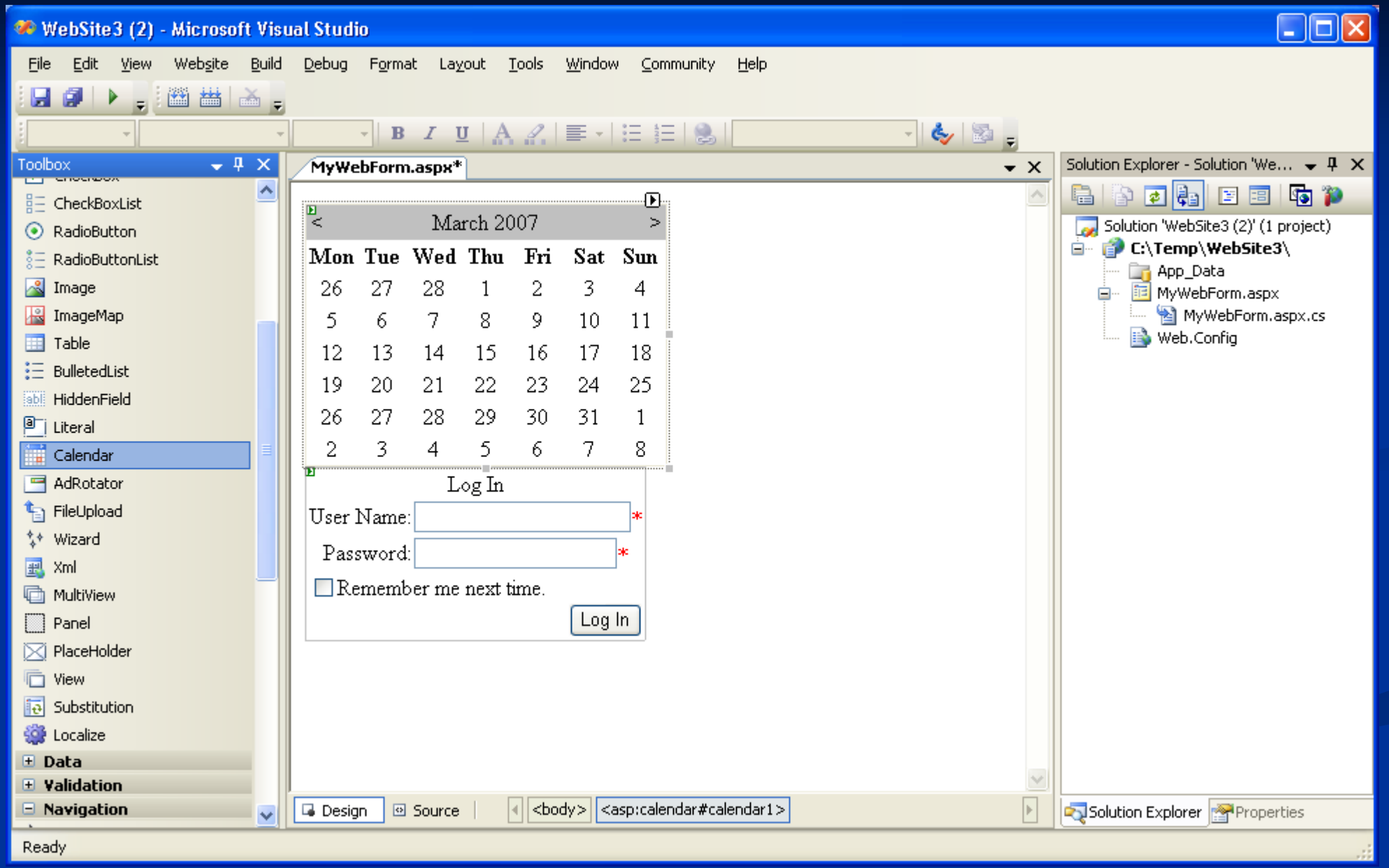
Text:	"Increment"
EventClick:	Butt1_Click
...	

```
HTTP Response
<html>
<input ViewState ...
   value="jkl43kls"/>
  (TextBox1.Text="20")
<input TextBox1 ...
    value="20" />
<input Button1 .../>
</html>
```

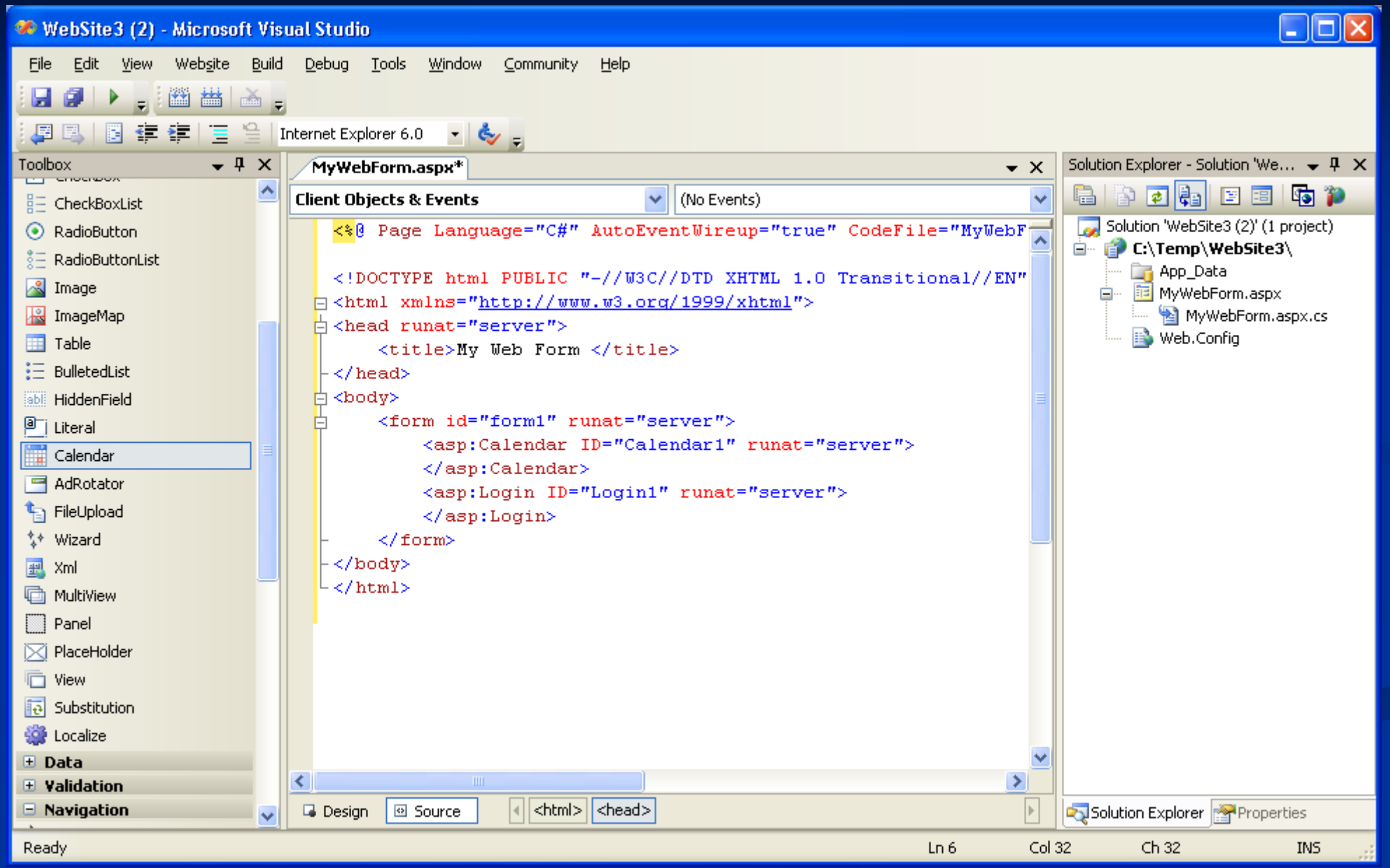
Step 7. Process Posted Events

(IsPostBack = true)

Exotic Web Controls



Exotic Web Controls

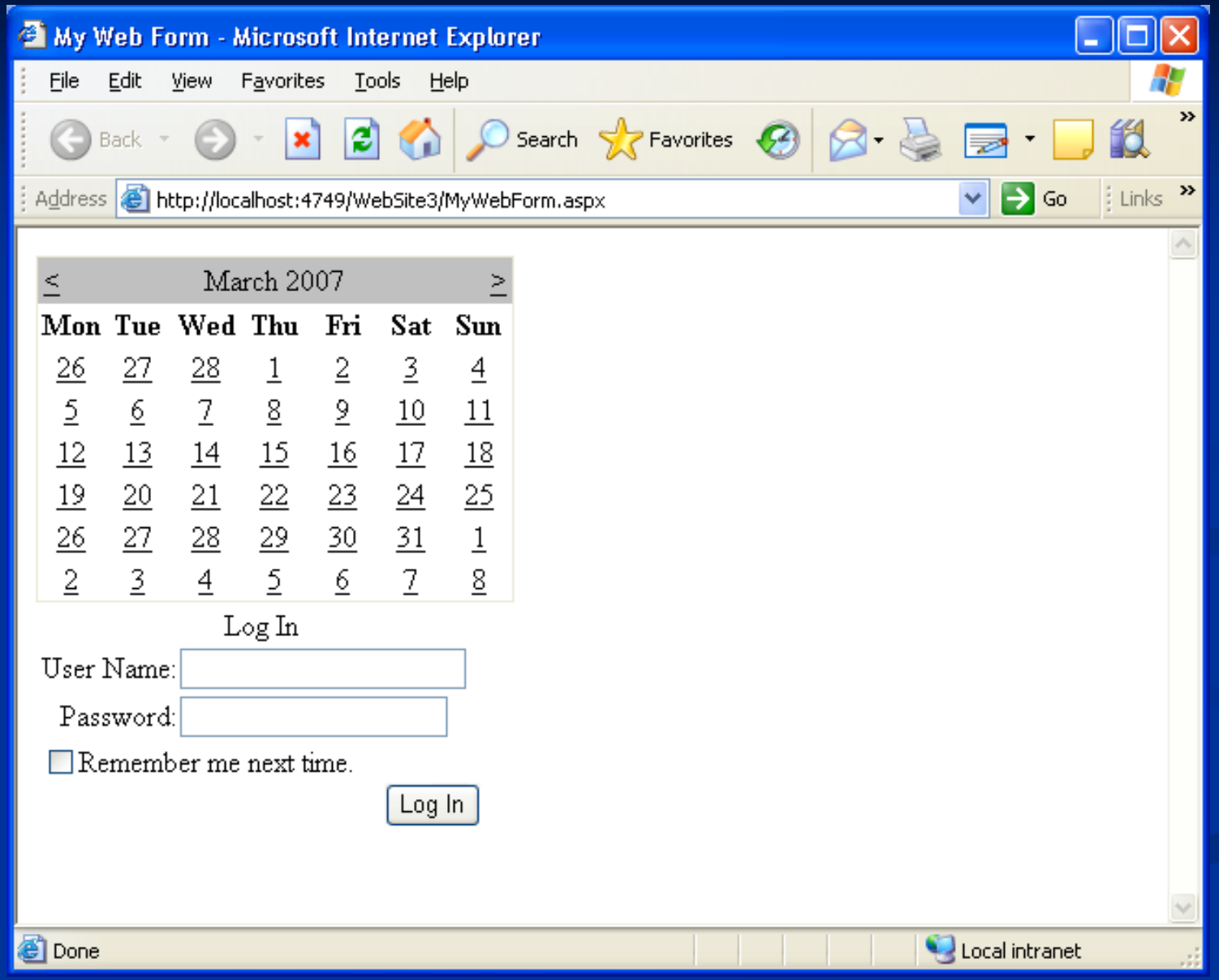


Exotic Web Controls Render as HTML

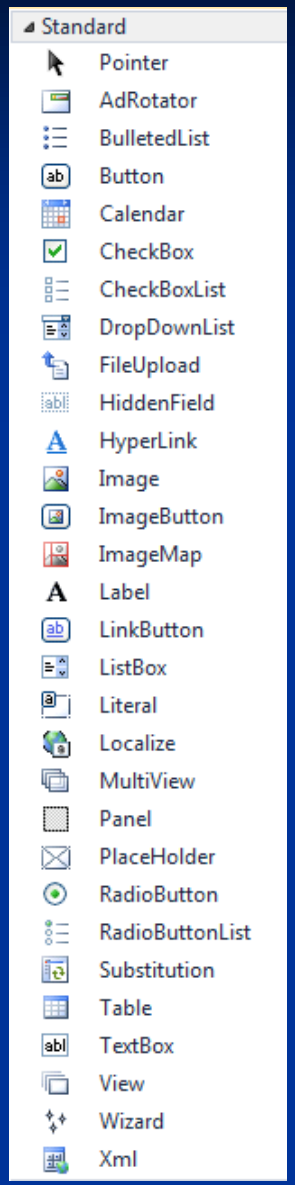
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>My Web Form </title>
</head>
<body>
    <form name="form1" method="post" action="MyWebForm.aspx" onsubmit="javascript:return WebForm_OnSubmit();"
        id="form1">
        <div>...</div>

        <script>...</script>

        <script src="/WebSite3/WebResource.axd?d=xb5Sqa2qVM2zhXRnpvfKrQ2&amp;t=6329691054018535 48"
            type="text/javascript"></script>

        <script src="/WebSite3/WebResource.axd?d=3TfefCACbiOgx_BzAxl-P1Ini3iBGBYaeCK_ZAnOpu81&amp;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
                                    &lt;</a></td>
                            <td align="center" style="width: 70%;">
                                March 2007</td>
```

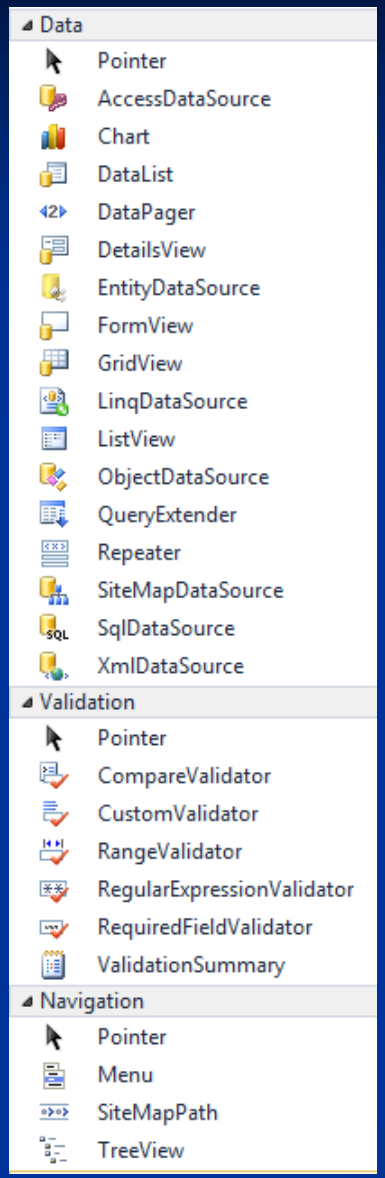
Design | Source | <html> | <body> | <form#form1> | <script>

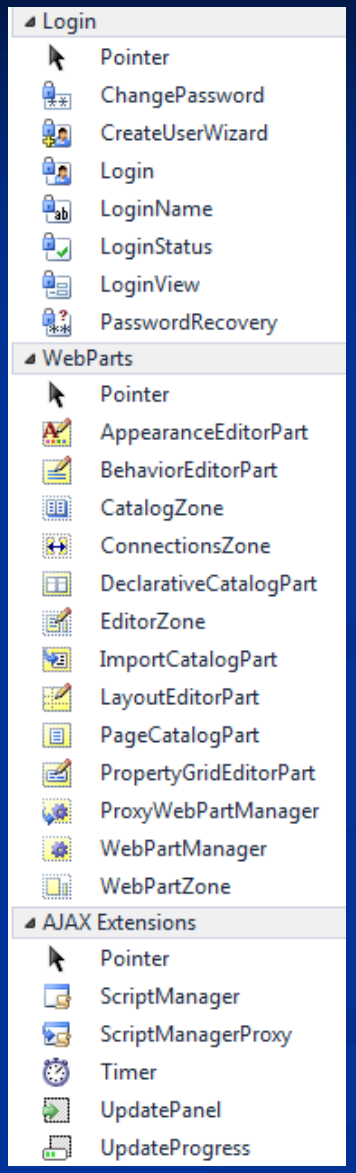
Exotic Web Controls Render as HTML

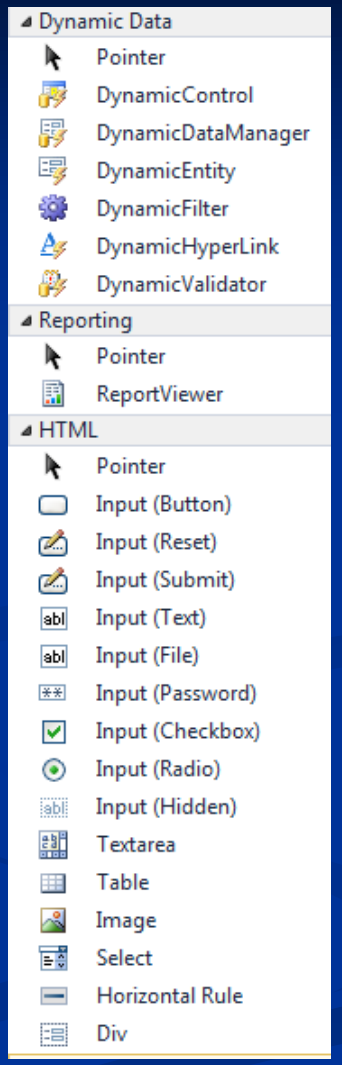


Web Controls









Data Binding

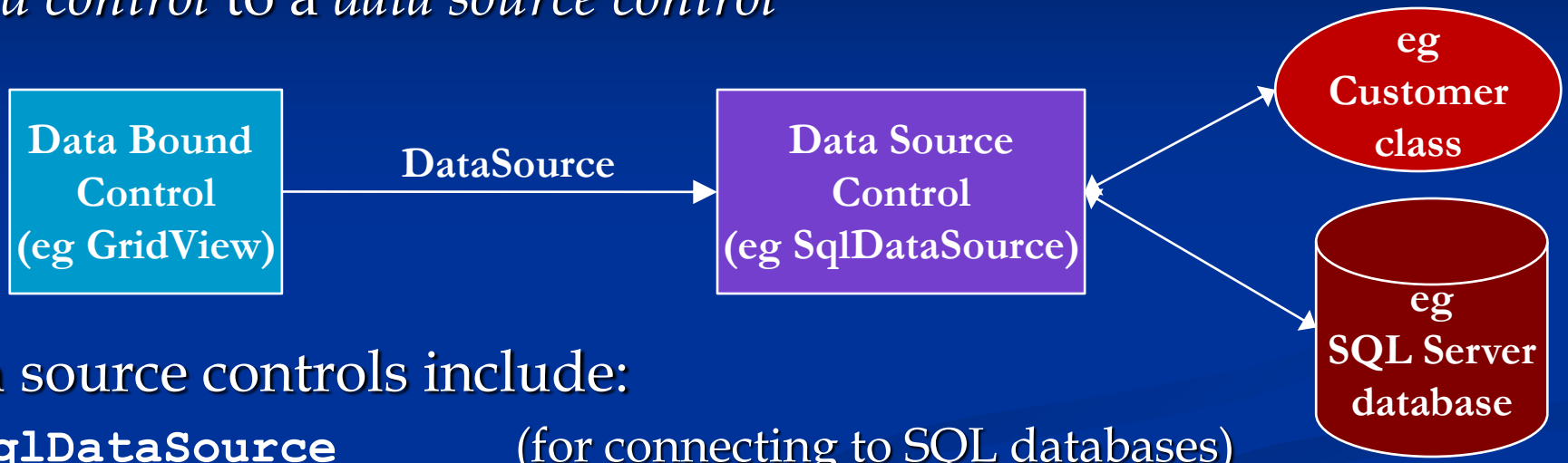
- Many simple web controls such as Buttons, Labels and Textboxes have a property called Text that you can get and set.

```
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*.

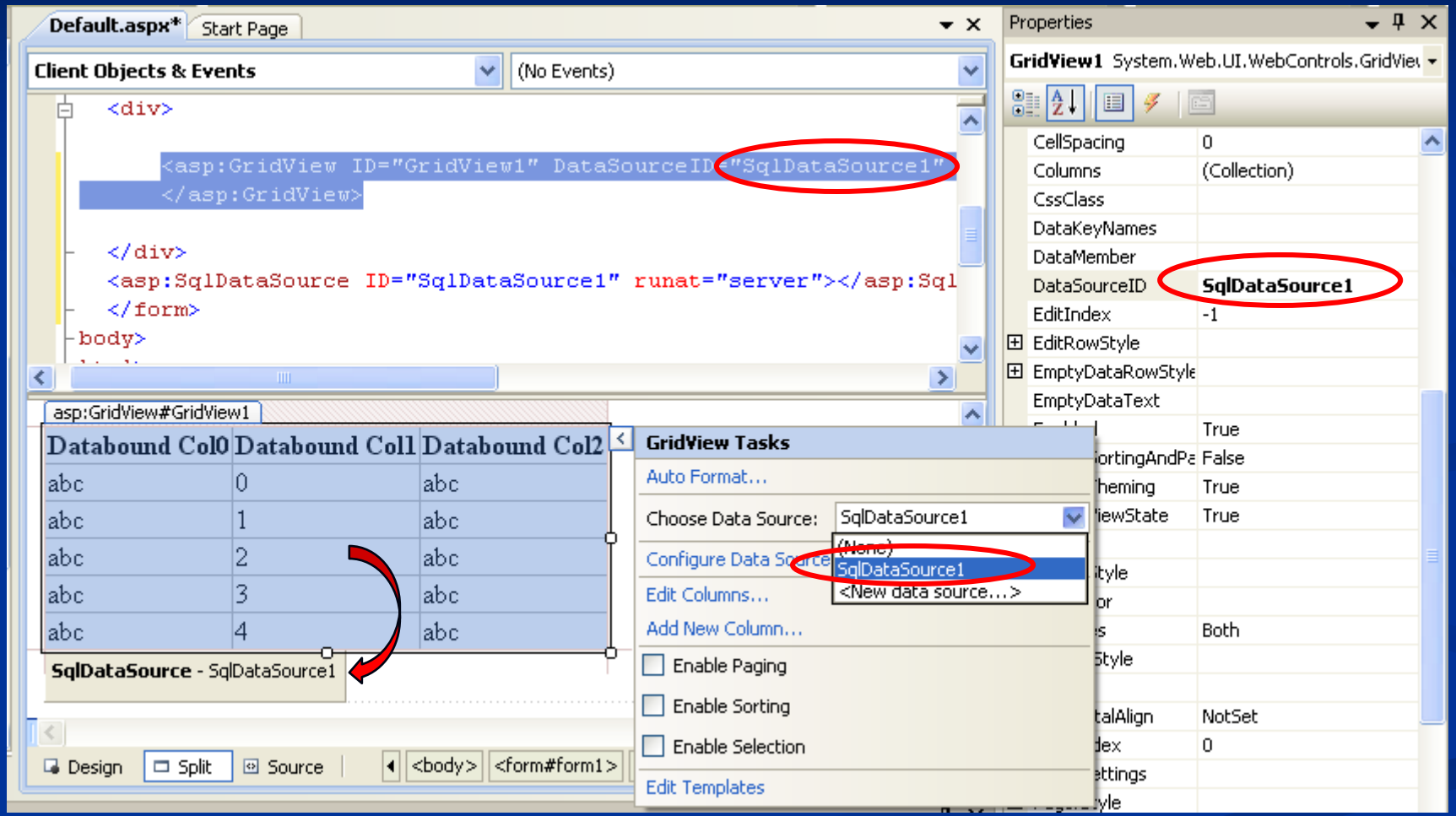
Declarative Data Binding

- 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.

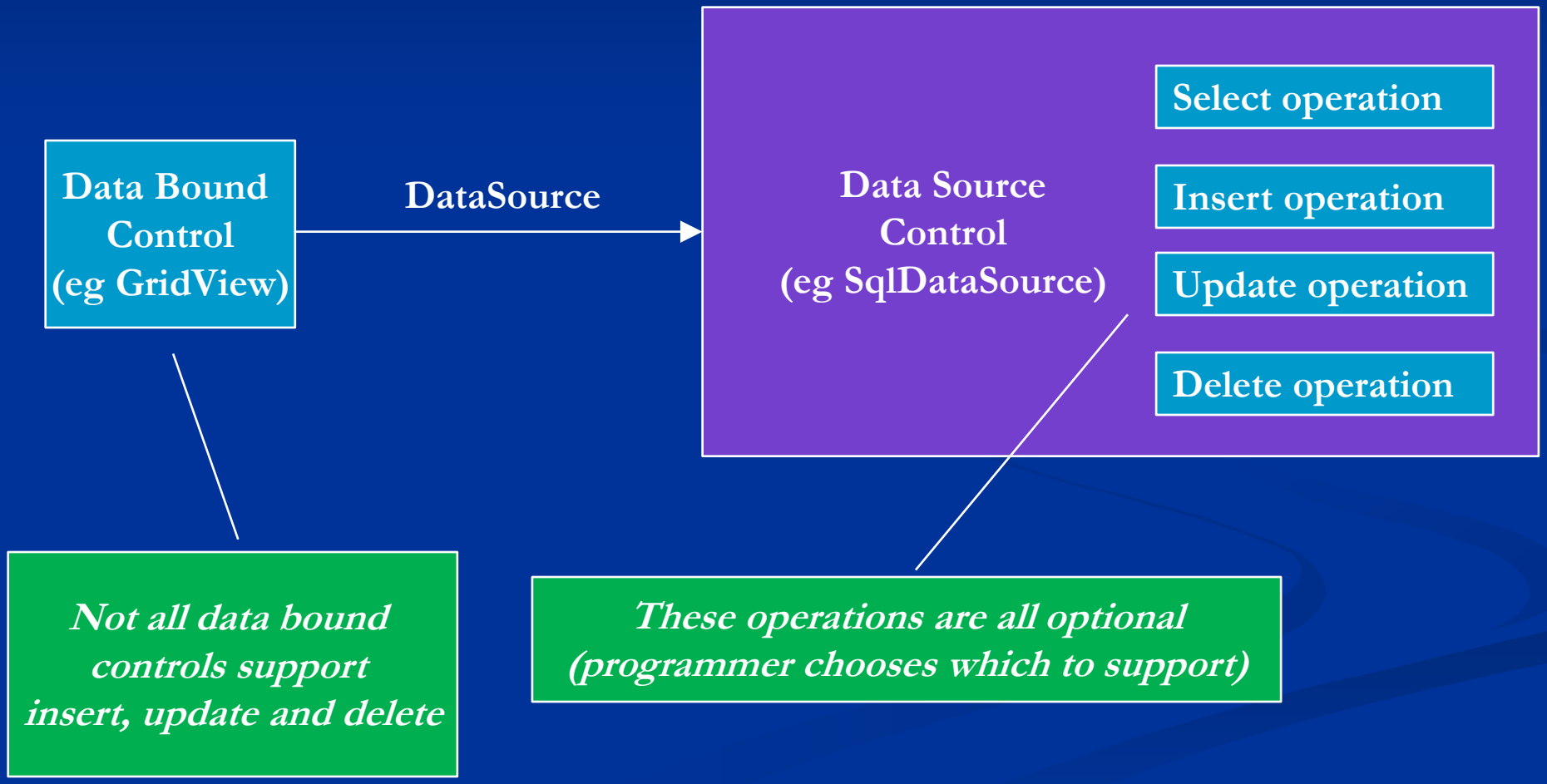
Configuring DataSourceID



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]);
}
```

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

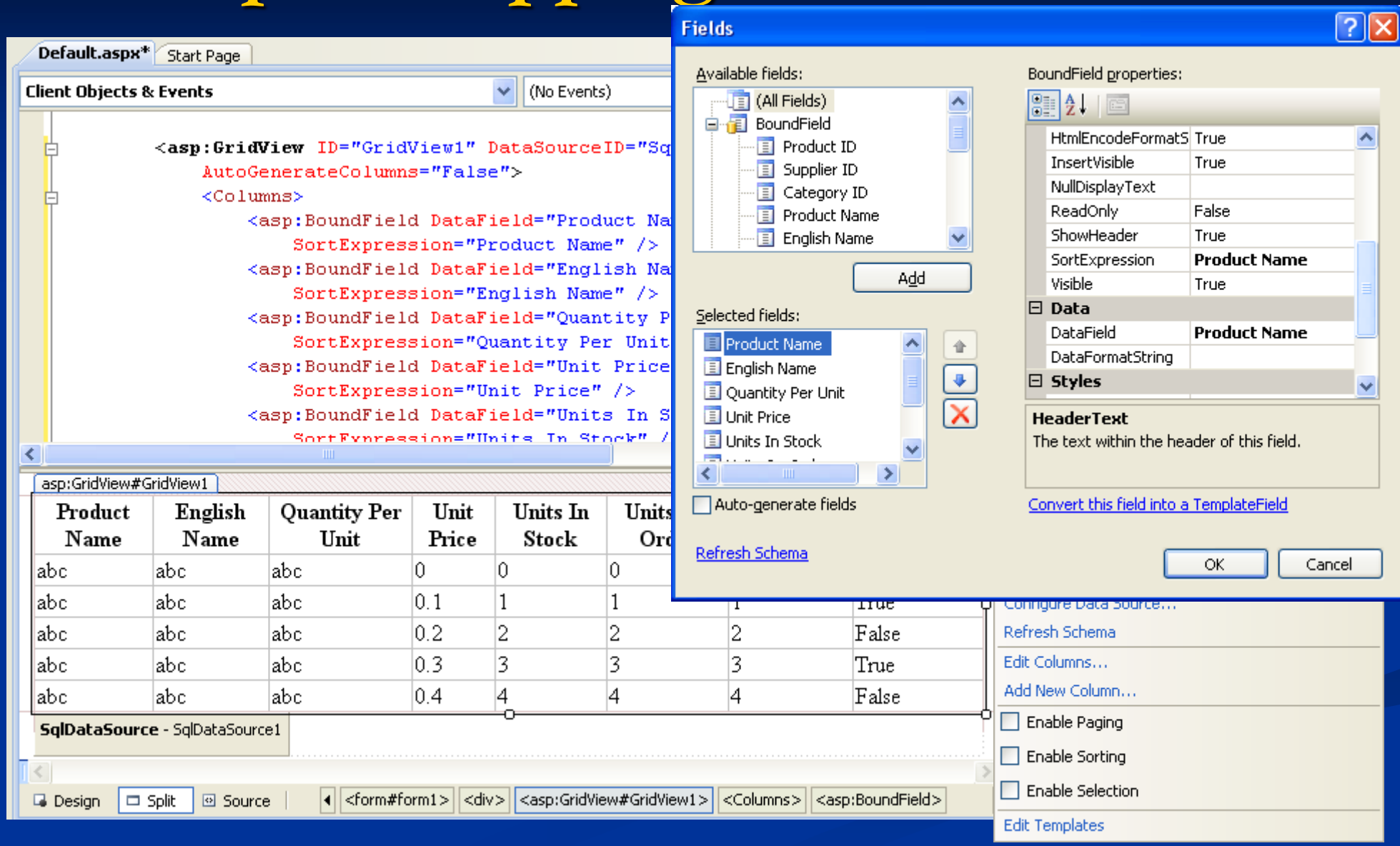
Event handler for RowDeleted event on GridView1

Do the actual data binding now!

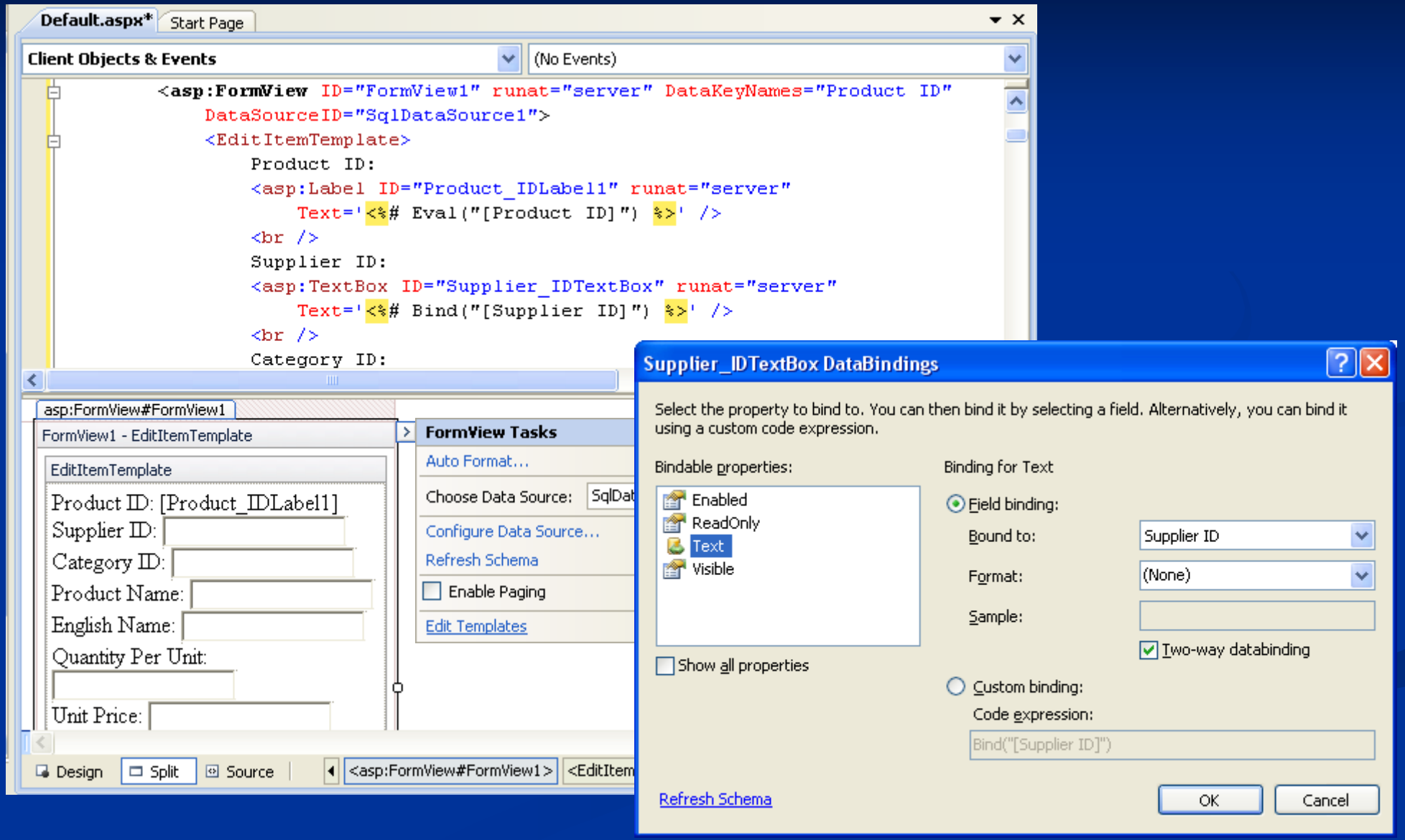
Mapping Data Bound Fields

- 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



Example: Mapping FormView fields



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.