MS ASP.NET

2007/2008

http://samples.gotdotnet.com/quickstart/aspplus/

Cechy

- Środowisko tworzenia aplikacji WWW
  - kod kompilowany
  - oparte o Common Language Runtime
  - konfigurowalne za pomocą plików tekstowych
  - zoptymalizowane pod kątem systemów wieloprocesorowych
  - łatwość rozszerzania
  - bezpieczeństwo
  - wsparcie dla C#, VB i JS#

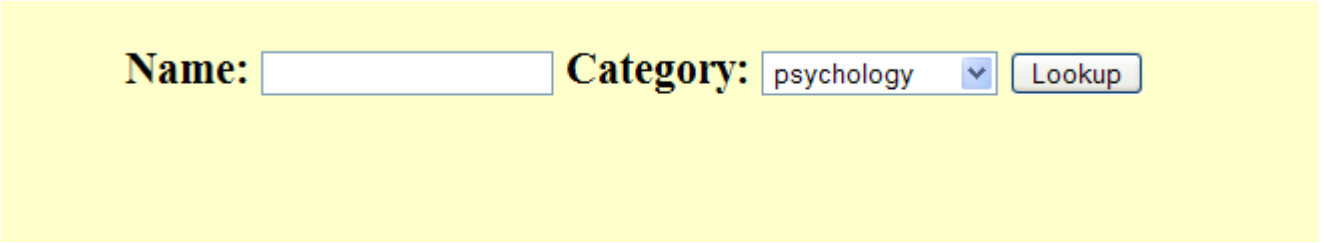
Web Forms

- Podstawę stanowią Web Forms:
  - rozwinięcie koncepcji ASP
  - umożliwiają tworzenie wielożywalnych kontrolek
  - opisują stronę w przejrzysty, logiczny sposób
  - umożliwiają pracę z edytorami WYSIWYG

Web Forms - cd

- pliki tekstowe z rozszerzeniem .aspx, serwowane przez IIS
- w momencie odwołania ze strony przeglądarki kompilowane do postaci klasy .NET
- najprostsza strona może zostać skonstruowana przez zmianę rozszerzenia pliku HTML na .aspx

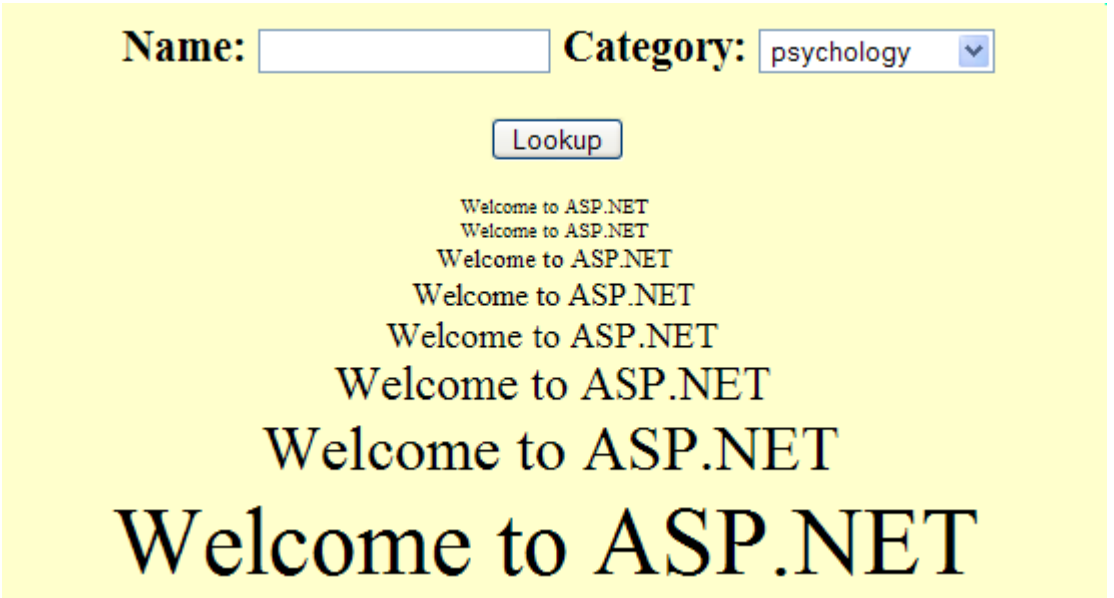
```
<html>
    <head>
        <link rel="stylesheet" href="intro.css">
    </head>

    <body>
        <center>
        <form action="intro1.aspx" method="post">
            <h3> Name: <input id="Name" type=text>
            Category:  <select id="Category" size=1>
                            <option>psychology</option>
                            <option>business</option>
                            <option>popular_comp</option>
                       </select>
            <input type=submit value="Lookup">
            </h3>
        </form>
        </center>
    </body>
</html>
```



Bloki <% %>

- Bloki ograniczone <% %> mogą być mieszane z kodem w HTMLu, zawierają fragmenty kodu kompilowane przez serwer

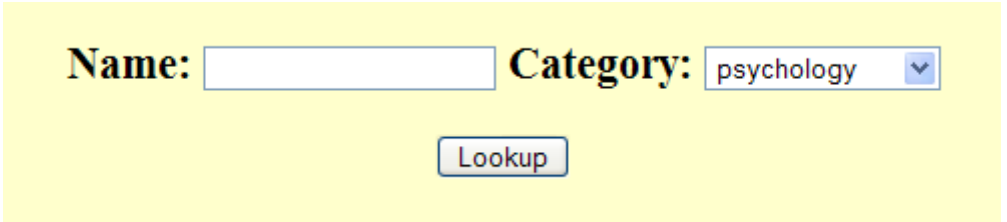
```
<%@ Page Language="C#"%>

<html>
   <head>
      <link rel="stylesheet"href="intro.css">
   </head>

   <body>
       <center>
       <form action="intro2.aspx" method="post">
           <h3> Name: <input id="Name" type=text>

           Category:  <select id="Category" size=1>
                          <option>psychology</option>
                          <option>business</option>
                          <option>popular_comp</option>
                      </select>
           </h3>
           <input type=submit value="Lookup">
           <p>
           <% for (int i=0; i <8; i++) { %>
              <font size="<%=i%>"> Welcome to ASP.NET </font> <br>
           <% }%>
       </form>
       </center>
   </body>
</html>
```



Kontrolki serwerowe

- Kontrolki serwerowe stanowią elegancką alternatywę dla bloków <% %>
- Są deklarowane przy pomocy specjalnych tagów lub tagów HTML z atrybutem `runat=”server”`
- Zawarte są w przestrzeni System.Web.UI.HtmlControls i System.Web.UI.WebControls

```
<%@ Page Language="C#"%>

<html>
    <head>
       <link rel="stylesheet"href="intro.css">
    </head>

    <body>
        <center>
        <form action="intro4.aspx" method="post" runat=server>
            <h3> Name: <asp:textbox id="Name" runat="server"/>
            Category:  <asp:dropdownlist id="Category" runat=server>
                          <asp:listitem>psychology</asp:listitem>
                          <asp:listitem>business</asp:listitem>
                          <asp:listitem>popular_comp</asp:listitem>
                       </asp:dropdownlist>
            </h3>
            <asp:button text="Lookup" runat="server"/>
        </form>
        </center>
    </body>
</html>
```



Rodzaje kontrolek serwerowych

- kontrolki serwerowe HTML – elementy HTML dostępne dla aplikacji działającej na serwerze. Model obiektowy blisko powiązany z elementem HTML
- kontrolki Web – większa funkcjonalność, bardziej abstrakcyjne
- kontrolki sprawdzające – umożliwiają testowanie danych wprowadzonych przez użytkownika
- kontrolki użytkownika – tworzone jako strony zawierające inne kontrolki

Kontrolki HTML

- elementy HTML zawierające atrybuty dzięki którym są widziane przez serwer (domyślnie elementy HTML nie są widziane przez serwer)
- każdy element HTML może być przekształcony w kontrolkę poprzez dodanie atrybutu runat=”server” oraz atrybutu ID
- kontrolka jest widziana jako obiekt
- dziedziczy z System.Web.UI.HtmlControls.HtmlControl

```
<html>
    <script language="C#" runat="server">

        void Page_Load(Object Src, EventArgs E) {
            Message.Text = "You last accessed this page at: " +
              DateTime.Now;
        }

    </script>

    <body>
       <h3><font face="Verdana">Manipulating Server Controls</font></h3>

       This sample demonstrates how to manipulate
       the &lt;asp:label&gt; server control within
       the Page_Load event to output the current time.

       <p>
       <hr>

       <asp:label id="Message" font-size="24" font-bold="true"
           runat=server/>

    </body>
</html>
```

Cechy kontrolek HTML

- obsługują zdarzenia na serwerze
- mogą obsługiwać zdarzenia po stronie klienta
- zachowują swój stan przy przesłaniach do/z serwera
- mogą współpracować z kontrolkami sprawdzającymi
- wparcie dla styli (CSS) – o ile przeglądarka klienta je obsługuje
- możliwość dodawania własnych atrybutów

```
<html>
    <script language="C#" runat="server">
        void EnterBtn_Click(Object Src, EventArgs E) {
            Message.Text = "Hi " + Name.Text + ", welcome to ASP.NET!";
        }
    </script>

    <body>
        <h3><font face="Verdana">Handling Control Action Events</font></h
        <p>
        This sample demonstrates how to access a &lt;asp:textbox&gt;
        server control within the "Click" event of a &lt;asp:button&gt;,
        and use its content to modify the text of a &lt;asp:label&gt;.
        <p>
        <hr>

        <form action="controls3.aspx" runat=server>
            <font face="Verdana">
                Please enter your name:
                    <asp:textbox id="Name" runat=server/>
                    <asp:button text="Enter" Onclick="EnterBtn_Click"
                        runat=server/>

                <p>
                <asp:label id="Message"  runat=server/>

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

Handling Control Action Events

This sample demonstrates how to access a <asp:textbox> server control within the "Click" event of a <asp:button>, and use its content to modify the text of a <asp:label>.

Please enter your name: John Enter

Hi John, welcome to ASP.NET!

```
<%@ Page Language="C#" %>
<html>
<body>-->
  <h3><font face="verdana">Applying Styles to HTML Controls</font></h3>
  <p><font face="verdana"><h4>Styled Span</h4></font><p>
  <span style="font: 12pt verdana; color:orange;font-weight:700"
   runat="server">This is some literal text inside a styled span control
  </span>

  <p><font face="verdana"><h4>Styled Button</h4></font><p>
  <button style="font: 8pt verdana;background-color:lightgreen;
   border-color:black;width:100" runat="server">Click me!</button>

  <p><font face="verdana"><h4>Styled Text Input</h4></font><p>

  Enter some text: <p>
  <input type="text" value="One, Two,
    Three" style="font: 14pt verdana;
    background-color:yellow;
    border-style:dashed;
    border-color:red;
    width:300;" runat="server"/>

  <p><font face="verdana"><h4>
    Styled Select Input</h4></font><p>

  Select an item: <p>
  <select style="font: 14pt verdana;
   background-color:lightblue;
   color:purple;" runat="server">
    <option>Item 1</option>
    <option>Item 2</option>
    <option>Item 3</option>
  </select>
</body>
</html>
```

Applying Styles to HTML Controls

Styled Span

This is some literal text inside a styled span control

Styled Button

Click me!

Styled Text Input

Enter some text:

One, Two, Three

Styled Select Input

Select an item:

Item 1

Dodawanie kontrolek HTML do strony

- kontrolki HTML można dodawać:
  - graficznie przy pomocy edytora
  - bezpośrednio w kodzie HTML
  - programowo:
    - Umieścić na stronie kontrolkę PlaceHolder lub Panel
    - utworzyć obiekt pożądanej kontrolki i ustawić jej własności
    - dodać do kolekcji Controls kontrolki PlaceHolder lub Panel
    - aczkolwiek można dodawać kontrolkę w dowolnym miejscu kolekcji, jedynie dodanie jej na końcu jest bezproblemowe

```
<body>
   <form runat="server">
      <h3>PlaceHolder Example</h3>

      <asp:PlaceHolder id="PlaceHolder1"
           runat="server"/>
   </form>
</body>
```

```
private void DropDownList1_SelectedIndexChanged(object sender,
    System.EventArgs e)
{
    // Get the number of labels to create.
    int numlabels = System.Convert.ToInt32(DropDownList1.
        SelectedItem.Text);
    for (int i=1; i<=numlabels; i++)
    {
        Label myLabel = new Label();
        // Set the label's Text and ID properties.
        myLabel.Text = "Label" + i.ToString();
        myLabel.ID = "Label" + i.ToString();
        PlaceHolder1.Controls.Add(myLabel);
        // Add a spacer in the form of an HTML <BR> element.
        PlaceHolder1.Controls.Add(new LiteralControl("<br>"));
    }
}
```

System.Web.UI.Control

- System.Object
  System.Web.UI.Control

- Własności:
  - `public virtual ControlCollection Controls {get;}`
  - `protected virtual HttpContext Context {get;}`
  - `protected EventHandlerList Events {get;}`
  - `public virtual string ID {get; set;}`
  - `public virtual Page Page {get; set;}`
  - `public virtual Control Parent {get;}`
  - `public virtual bool Visible {get; set;}`

```
private void Button1_Click(object sender, EventArgs MyEventArgs)
{
      // Find control on page.
      Control myControl1 = FindControl("TextBox2");
      if(myControl1!=null)
      {
         // Get control's parent.
         Control myControl2 = myControl1.Parent;
         Response.Write("Parent of the text box is : " + myControl2.ID);
      }
      else
      {
         Response.Write("Control not found");
      }
}
```

System.Web.UI.Control - cd

- Metody:
  - `protected virtual object SaveViewState();`
  - `protected virtual void LoadViewState(object savedState);`
  - `protected virtual void Render(HtmlTextWriter writer);`
  - `protected virtual void OnInit(EventArgs e);`
  - `protected virtual void OnLoad(EventArgs e);`
  - `protected virtual void OnUnload(EventArgs e);`
- Zdarzenia:
  - `public event EventHandler Init;`
  - `public event EventHandler Load;`
  - `public event EventHandler Unload;`

```
<%@ Page language="c#" Codebehind="WebForm1.aspx.cs"
 AutoEventWireup="false" Inherits="WebApplication5.WebForm1" %>
<%@ Register TagPrefix="UserCtrlSample" Namespace="WebApplication5"
 Assembly="WebApplication5" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" >
<HTML>
    <HEAD>
        <title>WebForm1</title>
        <meta content="Microsoft Visual Studio .NET 7.1"
          name="GENERATOR">
        <meta content="C#" name="CODE_LANGUAGE">
        <meta content="JavaScript" name="vs_defaultClientScript">
        <meta content="http://schemas.microsoft.com/intellisense/ie5"
          name="vs_targetSchema">
    </HEAD>
    <body MS_POSITIONING="GridLayout">
        <form id="Form1" method="post" runat="server">
            <USERCTRLSAMPLE:MYCTRL id="Control1" runat="server">
             </USERCTRLSAMPLE:MYCTRL></form>
    </body>
</HTML>
```

```
namespace WebApplication5{
    public class WebForm1 : System.Web.UI.Page{
        protected WebApplication5.MyCtrl Control1;
        TextWriter myFile = File.CreateText
            (@"C:\Inetpub\wwwroot\NewTextFile.txt");
        private void Page_Load(object sender, System.EventArgs e){
            Control1.file = myFile;
            myFile.WriteLine("Page has loaded.");}
        private void Page_Unload(object sender, System.EventArgs e)    {
            myFile.WriteLine("Page was unloaded.");
            myFile.Close();}
        override protected void OnInit(EventArgs e){
            InitializeComponent();
            base.OnInit(e);}
        private void InitializeComponent(){
            this.Load += new System.EventHandler(this.Page_Load);
            this.Unload += new System.EventHandler(this.Page_Unload);}
    }
    public class MyCtrl : Control{
        public TextWriter file;
        public MyCtrl(){
            Load += new EventHandler(MyControl_Load);
            Unload += new EventHandler(MyControl_Unload);}
        protected override void Render(HtmlTextWriter output){
            output.Write("{0} {1} {2}","<H2>",
            "Welcome to Control Development!", "</H2>");}
        void MyControl_Load(object sender,EventArgs e){
            file.WriteLine("Custom control has loaded.");}
        void MyControl_Unload(object sender,EventArgs e){
            file.WriteLine("Custom control was unloaded.");}
    }
}
```

```
Page has loaded.
Custom control has loaded.
Custom control was unloaded.
Page was unloaded.
```

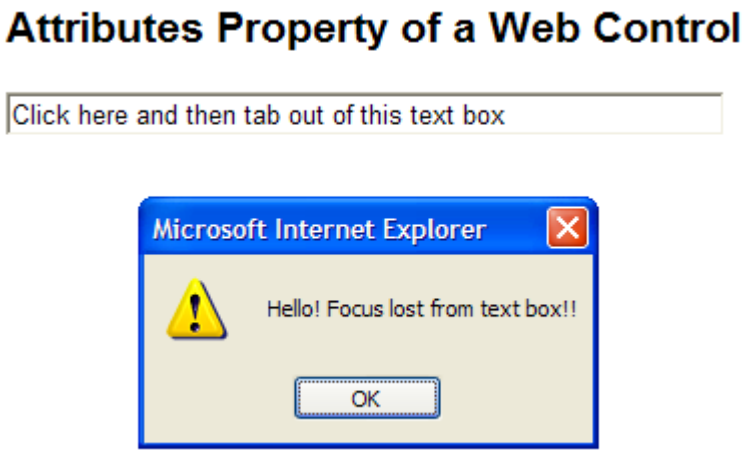
Welcome to Control Development!

```
protected override object SaveViewState()
{  // Change Text Property of Label when this function is invoked.
   if(HasControls() && (Page.IsPostBack))
   {
      ((Label)(Controls[0])).Text = "Custom Control Has Saved State";
   }
   // Save State as a cumulative array of objects.
   object baseState = base.SaveViewState();
   string userText = UserText;
   string passwordText = PasswordText;
   object[] allStates = new object[3];
   allStates[0] = baseState;
   allStates[1] = userText;
   allStates[2] = PasswordText;
   return allStates;
}

 protected override void LoadViewState(object savedState)
 {
    if (savedState != null)
    {
       // Load State from the array of objects that was saved at ;
       // SavedViewState.
       object[] myState = (object[])savedState;
       if (myState[0] != null)
          base.LoadViewState(myState[0]);
       if (myState[1] != null)
          UserText = (string)myState[1];
       if (myState[2] != null)
          PasswordText = (string)myState[2];
    }
 }
```

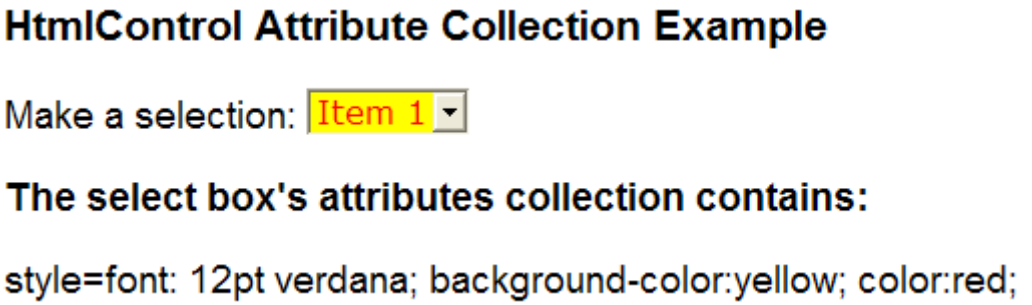
System.Web.UI.HtmlControls.HtmlControl

- System.Object
  System.Web.UI.Control
    System.Web.UI.HtmlControls.HtmlControl
      System.Web.UI.HtmlControls.HtmlContainerControl
      System.Web.UI.HtmlControls.HtmlImage
      System.Web.UI.HtmlControls.HtmlInputControl
- `public abstract class HtmlControl : Control, IAttributeAccessor`
- Własności:
  - `public AttributeCollection Attributes {get;}`
  - `public bool Disabled {get; set;}`
  - `public CssStyleCollection Style {get;}`
  - `public virtual string TagName {get;}`

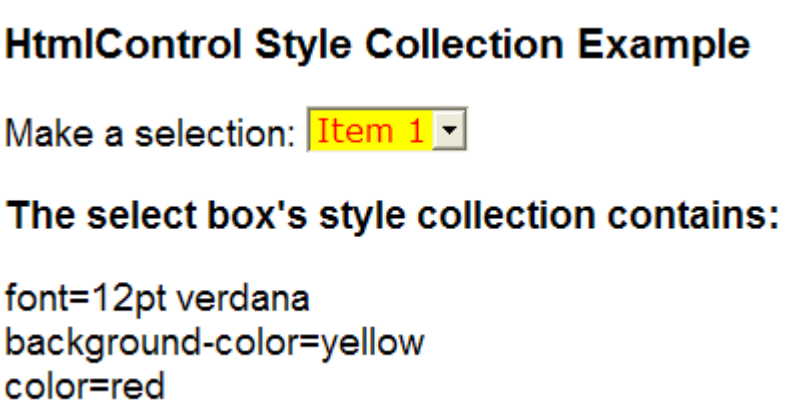
```
<html>
<head>
   <script language="C#" runat="server">

       void Page_Load(Object sender, EventArgs e) {
          TextBox1.Attributes["onblur"]="javascript:alert('Hello!
            Focus lost from text box!!');";
       }
   </script>

</head>
<body>
   <h3>Attributes Property of a Web Control</h3>
<form runat="server">

   <asp:TextBox id="TextBox1" columns=54
    Text="Click here and then tab out of this text box"
    runat="server"/>

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



```
<%@ Page Language="C#" AutoEventWireup="True" %>
<html>
<script language="C#" runat="server">
   void Page_Load(Object sender, EventArgs e){
      Message.InnerHtml = "<h4>The select box's attributes collection
       contains:</h4>";
      IEnumerator keys = Select.Attributes.Keys.GetEnumerator();
      while (keys.MoveNext()){
         String key = (String)keys.Current;
         Message.InnerHtml += key +"="+ Select.Attributes[key] + "<br>";
      }
   }
</script>

<body>
   <h3>HtmlControl Attribute Collection Example</h3>
   Make a selection:
   <select id="Select"
           style="font: 12pt verdana;
                  background-color:yellow;
                  color:red;"
           runat="server">
      <option>Item 1</option>
      <option>Item 2</option>
      <option>Item 3</option>
   </select>
   <p>
   <span id="Message" MaintainState="false" runat="server" />
</body>
</html>
```



```
<%@ Page Language="C#" AutoEventWireup="True" %>
<html>
<script language="C#" runat="server">
   void Page_Load(Object sender, EventArgs e){
      Message.InnerHtml = "<h4>The select box's style collection
       contains:</h4>";
      IEnumerator keys = Select.Style.Keys.GetEnumerator();
      while (keys.MoveNext()){
         String key = (String)keys.Current;
         Message.InnerHtml += key +"="+ Select.Style[key] + "<br>";
      }
   }
</script>

<body>
   <h3>HtmlControl Style Collection Example</h3>
   Make a selection:
   <select id="Select"
           style="font: 12pt verdana;
                  background-color:yellow;
                  color:red;"
           runat="server">
      <option>Item 1</option>
      <option>Item 2</option>
      <option>Item 3</option>
   </select>
   <p>
   <span id="Message" MaintainState="false" runat="server" />
</body>
</html>
```



- HtmlAnchor <a>
- HtmlButton <button>
- HtmlForm <form>
- HtmlGenericControl
- HtmlImage <img>
- HtmlInputButton <input type= button>, <input type= submit>, <input type= reset>
- HtmlInputCheckBox <input type= checkbox>
- HtmlInputFile <input type= file>
- HtmlInputHidden <input type=hidden>
- HtmlInputImage <input type= image>
- HtmlInputRadioButton <input type= radio>
- HtmlInputText <input type= text>, <input type= password>
- HtmlSelect <select>
- HtmlTable <table>
- HtmlTableCell <td>, <th> w HtmlTableRow.
- HtmlTableCellCollection
- HtmlTableRow <tr>
- HtmlTableRowCollection
- HtmlTextArea <textarea>

Kontrolki Web

- Większe możliwości
- Automatyczna detekcja przeglądarki
- Niektóre umożliwiają definiowanie wyglądu przy pomocy wzorców (templates)
- Niektóre umożliwiają opóźnienie przesłania informacji o zdarzeniu do serwera (i przesłania zdarzeń “w grupie”)
- Możliwość przekazywania zdarzeń od elementów składowych do elementu nadrzędnego (np. z przycisku w tabeli – do tabeli)

```
<html>
  <head>
    <script language="C#" runat="server">
      void Button1_Click(object Source, EventArgs e){
          String s = "Selected items:<br>";
          for (int i=0; i < Check1.Items.Count; i++){
              if ( Check1.Items[ i ].Selected ){
                  s = s + Check1.Items[i].Text;
                  s = s + "<br>";
              }
          }
          Label1.Text = s;
      }
      void chkLayout_CheckedChanged(Object sender, EventArgs e){
          if (chkLayout.Checked == true){
              Check1.RepeatLayout = RepeatLayout.Table;
          }else {
              Check1.RepeatLayout = RepeatLayout.Flow;
          }
      }
      void chkDirection_CheckedChanged(Object sender, EventArgs e) {
          if (chkDirection.Checked == true) {
              Check1.RepeatDirection = RepeatDirection.Horizontal;
          }else{
              Check1.RepeatDirection = RepeatDirection.Vertical;
          }
      }
    </script>
  </head>
```

```
<body>
    <h3><font face="Verdana">CheckBoxList Example</font></h3>
    <form runat=server>
        <asp:CheckBoxList id=Check1 runat="server">
            <asp:ListItem>Item 1</asp:ListItem>
            <asp:ListItem>Item 2</asp:ListItem>
            <asp:ListItem>Item 3</asp:ListItem>
            <asp:ListItem>Item 4</asp:ListItem>
            <asp:ListItem>Item 5</asp:ListItem>
            <asp:ListItem>Item 6</asp:ListItem>
        </asp:CheckBoxList>
        <p>
        <asp:CheckBox id=chkLayout
         OnCheckedChanged="chkLayout_CheckedChanged"
         Text="Display Table Layout" Checked=true AutoPostBack="true"
         runat="server" />
        <br>
        <asp:CheckBox id=chkDirection
         OnCheckedChanged="chkDirection_CheckedChanged"
         Text="Display Horizontally" AutoPostBack="true"
         runat="server" />
        <p>
        <asp:Button id=Button1 Text="Submit" onclick="Button1_Click"
         runat="server"/>
        <p>
        <asp:Label id=Label1 font-name="Verdana" font-size="8pt"
         runat="server"/>
    </form>
</body>
</html>
```

CheckBoxList Example

- [x] Item 1
- [ ] Item 2
- [x] Item 3
- [ ] Item 4
- [ ] Item 5
- [ ] Item 6

- [x] Display Table Layout
- [ ] Display Horizontally

Submit

Selected items:
Item 1
Item 3

CheckBoxList Example

- [x] Item 1 [ ] Item 2 [x] Item 3 [ ] Item 4 [ ] Item 5 [ ] Item 6

- [x] Display Table Layout
- [x] Display Horizontally

Submit

Selected items:
Item 1
Item 3

```
<%@ Import Namespace="System.Data" %>

<html>
<script language="C#" runat="server">
    ICollection CreateDataSource() {
        DataTable dt = new DataTable();
        DataRow dr;
        dt.Columns.Add(new DataColumn("IntegerValue",typeof(Int32)));
        dt.Columns.Add(new DataColumn("StringValue",typeof(string)));
        dt.Columns.Add(new DataColumn("DateTimeValue",typeof(DateTime)));
        dt.Columns.Add(new DataColumn("BoolValue",typeof(bool)));
        dt.Columns.Add(new DataColumn("CurrencyValue",typeof(double)));
        for (int i = 0; i < 9; i++) {
            dr = dt.NewRow();
            dr[0] = i;
            dr[1] = "Item " + i.ToString();
            dr[2] = DateTime.Now;
            dr[3] = (i % 2 != 0) ? true : false;
            dr[4] = 1.23 * (i+1);
            dt.Rows.Add(dr);
        }
        DataView dv = new DataView(dt);
        return dv;
    }

    void Page_Load(Object sender, EventArgs e) {
        MyDataGrid.DataSource = CreateDataSource();
        MyDataGrid.DataBind();
    }

</script>
```

```
<body>

    <h3><font face="Verdana">Simple DataGrid Example</font></h3>

    <form runat=server>

      <ASP:DataGrid id="MyDataGrid" runat="server"
        BorderColor="black"
        BorderWidth="1"
        GridLines="Both"
        CellPadding="3"
        CellSpacing="0"
        Font-Name="Verdana"
        Font-Size="8pt"
        HeaderStyle-BackColor="#aaaadd"
      />

  </form>

</body>
</html>
```

Simple DataGrid Example

IntegerValue	StringValue	DateTimeValue	BoolValue	CurrencyValue
1	Item 1	12/17/2005 3:20:00 PM	True	2.46
2	Item 2	12/17/2005 3:20:00 PM	False	3.69
3	Item 3	12/17/2005 3:20:00 PM	True	4.92
4	Item 4	12/17/2005 3:20:00 PM	False	6.15
5	Item 5	12/17/2005 3:20:00 PM	True	7.38
6	Item 6	12/17/2005 3:20:00 PM	False	8.61
7	Item 7	12/17/2005 3:20:00 PM	True	9.84
8	Item 8	12/17/2005 3:20:00 PM	False	11.07
9	Item 9	12/17/2005 3:20:00 PM	True	12.3

Kontrolki sprawdzające

- HtmlInputText - Value
- HtmlTextArea - Value
- HtmlSelect - Value
- HtmlInputFile - Value
- TextBox - Text
- ListBox - SelectedItem.Value
- DropDownList - SelectedItem.Value
- RadioButtonList - SelectedItem.Value

```
<html>
<head>
    <script language="C#" runat=server>

        void ValidateBtn_Click(Object Sender, EventArgs E) {

            if (Page.IsValid == true) {
                lblOutput.Text = "Page is Valid!";
            }
            else {
                lblOutput.Text = "Some of the required fields are empty"
            }
        }

   </script>

</head>
```

```
<body>
<h3><font face="Verdana">Simple RequiredField Validator Sample</font>
</h3><p><form runat="server">
    <table bgcolor="#eeeeee" cellpadding=10>
    <tr valign="top"><td colspan=3>
      <asp:Label ID="lblOutput" Text="Fill in the required fields below"
       ForeColor="red" Font-Name="Verdana" Font-Size="10"
       runat=server /><br></td></tr>
    <tr><td colspan=3>
      <font face=Verdana size=2><b>Credit Card Information</b></font>
      </td></tr>
    <tr><td align=right>
        <font face=Verdana size=2>Card Type:</font>
      </td><td>
        <ASP:RadioButtonList id=RadioButtonList1 RepeatLayout="Flow"
             runat=server>
            <asp:ListItem>MasterCard</asp:ListItem>
            <asp:ListItem>Visa</asp:ListItem>
        </ASP:RadioButtonList>
      </td><td align=middle rowspan=1>
        <asp:RequiredFieldValidator id="RequiredFieldValidator1"
            ControlToValidate="RadioButtonList1"
            Display="Static"
            InitialValue="" Width="100%" runat=server>
            *
        </asp:RequiredFieldValidator>
      </td></tr>
    <tr><td align=right>
        <font face=Verdana size=2>Card Number:</font>
      </td><td>
        <ASP:TextBox id=TextBox1 runat=server />
      </td><td>
        <asp:RequiredFieldValidator id="RequiredFieldValidator2"
            ControlToValidate="TextBox1"
            Display="Static"
            Width="100%" runat=server>
            *
        </asp:RequiredFieldValidator>
      </td></tr>
```

```
<tr><td align=right>
        <font face=Verdana size=2>Expiration Date:</font>
      </td><td>
        <ASP:DropDownList id=DropDownList1 runat=server>
            <asp:ListItem></asp:ListItem>
            <asp:ListItem >06/00</asp:ListItem>
            <asp:ListItem >07/00</asp:ListItem>
            <asp:ListItem >08/00</asp:ListItem>
            <asp:ListItem >09/00</asp:ListItem>
            <asp:ListItem >10/00</asp:ListItem>
            <asp:ListItem >11/00</asp:ListItem>
            <asp:ListItem >01/01</asp:ListItem>
            <asp:ListItem >02/01</asp:ListItem>
            <asp:ListItem >03/01</asp:ListItem>
            <asp:ListItem >04/01</asp:ListItem>
            <asp:ListItem >05/01</asp:ListItem>
            <asp:ListItem >06/01</asp:ListItem>
            <asp:ListItem >07/01</asp:ListItem>
            <asp:ListItem >08/01</asp:ListItem>
            <asp:ListItem >09/01</asp:ListItem>
            <asp:ListItem >10/01</asp:ListItem>
            <asp:ListItem >11/01</asp:ListItem>
            <asp:ListItem >12/01</asp:ListItem>
        </ASP:DropDownList>
      </td><td>
        <asp:RequiredFieldValidator id="RequiredFieldValidator3"
          ControlToValidate="DropDownList1"
          Display="Static"
          InitialValue="" Width="100%" runat=server>
          *
        </asp:RequiredFieldValidator>
      </td><td></tr><tr><td></td><td>
        <ASP:Button id=Button1 text="Validate"
          OnClick="ValidateBtn_Click" runat=server />
      </td><td></td></tr></table>
</form>
</body>
</html>
```

Simple RequiredField Validator Sample

Some of the required fields are empty

Credit Card Information

Card Type: ○ MasterCard ○ Visa *

Card Number: *

Expiration Date: *

Validate

Simple RequiredField Validator Sample

Some of the required fields are empty

Credit Card Information

Card Type: ◉ MasterCard ○ Visa

Card Number: 123456

Expiration Date: *

Validate

Simple RequiredField Validator Sample

Page is Valid!

Credit Card Information

Card Type: ◉ MasterCard ○ Visa

Card Number: 123456

Expiration Date: 06/01

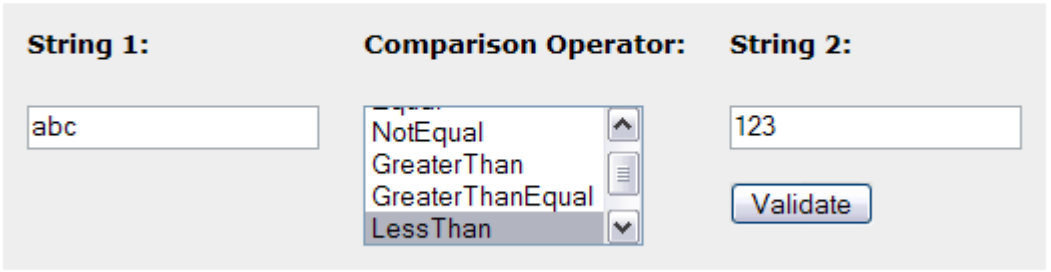
Validate

```
<%@ Page clienttarget=downlevel %>

<html>
<head>
    <script language="C#" runat="server">

        void Button1_OnSubmit(Object sender, EventArgs e) {

            if (Page.IsValid) {
               lblOutput.Text = "Result: Valid!";
            }
            else {
               lblOutput.Text = "Result: Not valid!";
            }
        }

        void lstOperator_SelectedIndexChanged(Object sender,
EventArgs e) {

            comp1.Operator = (ValidationCompareOperator)
lstOperator.SelectedIndex;
            comp1.Validate();
        }

   </script>

</head>
```

```
<body>
    <h3><font face="Verdana">CompareValidator Example</font></h3>
    <p>Type a value in each textbox, select a comparison operator,
       then click "Validate" to test.</p>
    <form runat=server><table bgcolor="#eeeeee" cellpadding=10>
      <tr valign="top"><td>
            <h5><font face="Verdana">String 1:</font></h5>
            <asp:TextBox id="txtComp" runat="server"></asp:TextBox>
        </td><td>
            <h5><font face="Verdana">Comparison Operator:</font></h5>
            <asp:ListBox id="lstOperator" OnSelectedIndexChanged=
             "lstOperator_SelectedIndexChanged" runat="server">
                    <asp:ListItem Selected Value="Equal" >Equal
                       </asp:ListItem>
                    <asp:ListItem Value="NotEqual" >NotEqual
                       </asp:ListItem>
                    <asp:ListItem Value="GreaterThan" >GreaterThan
                       </asp:ListItem>
                    <asp:ListItem Value="GreaterThanEqual" >
                       GreaterThanEqual</asp:ListItem>
                    <asp:ListItem Value="LessThan" >LessThan
                       </asp:ListItem>
                    <asp:ListItem Value="LessThanEqual" >LessThanEqual
                       </asp:ListItem>
            </asp:ListBox>
        </td><td>
            <h5><font face="Verdana">String 2:</font></h5>
            <asp:TextBox id="txtCompTo" runat="server"></asp:TextBox><p>
            <asp:Button runat=server Text="Validate" ID="Button1"
             onclick="Button1_OnSubmit" />
        </td></tr></table>
      <asp:CompareValidator id="comp1" ControlToValidate="txtComp"
          ControlToCompare = "txtCompTo" Type="String" runat="server"/>
      <br>
      <asp:Label ID="lblOutput" Font-Name="verdana" Font-Size="10pt"
          runat="server"/>
    </form>
</body>
</html>
```

CompareValidator Example

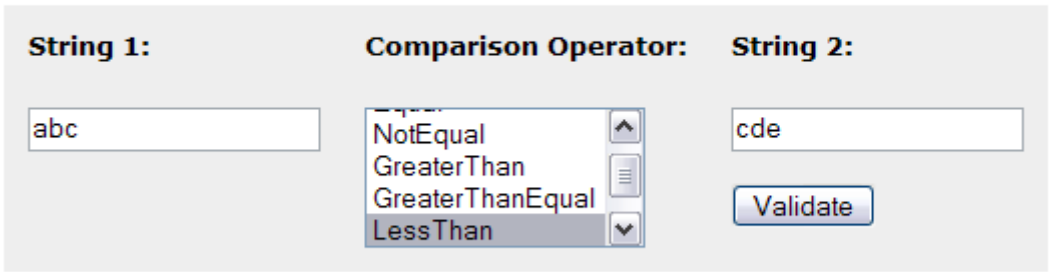
Type a value in each textbox, select a comparison operator, then click "Validate" to test.



Result: Not valid!

CompareValidator Example

Type a value in each textbox, select a comparison operator, then click "Validate" to test.



Result: Valid!

Inne rodzaje sprawdzania

- sprawdzanie zakresu
- sprawdzanie z użyciem wyrażeń regularnych
- sprawdzanie przy pomocy funkcji użytkownika

Kontrolki użytkownika

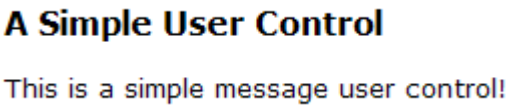
- Sposób na wielożywalność opracowanych stron ASP.NET
- Są typu `System.Web.UI.UserControl`
- Zazwyczaj zapisywane w pliku z rozszerzeniem ascx
- Włączane przy użyciu dyrektywy `Register`

Pagelet1.aspx

```
<%@ Page Language="C#" %>
<%@ Register TagPrefix="Acme" TagName="Message" Src="pagelet1.ascx" %>

<html>
<body style="font: 10pt verdana">

  <h3>A Simple User Control</h3>

  <Acme:Message runat="server"/>

</body>
</html>
```

Pagelet1.ascx

```
This is a simple message user control!
```



Pagelet2.aspx

```
<%@ Register TagPrefix="Acme" TagName="Message" Src="pagelet2.ascx" %>
<html>
  <script language="C#" runat="server">

      void SubmitBtn_Click(Object sender, EventArgs E) {
          MyMessage.Text = "Message text changed!";
          MyMessage.Color = "red";
      }

  </script>

<body style="font: 10pt verdana">
  <h3>A Simple User Control w/ Properties</h3>
  <form runat="server">
    <Acme:Message id="MyMessage" Text="This is a custom message!"
       Color="blue" runat="server"/><p>
    <asp:button text="Change Properties" OnClick="SubmitBtn_Click"
       runat=server/>
  </form>
</body>
</html>
```

Pagelet2.ascx

```
<script language="C#" runat="server">

  public String Color = "blue";
  public String Text = "This is a simple message user control!";

</script>

<span id="Message" style="color:<%=Color%>"><%=Text%></span>
```

A Simple User Control w/ Properties

This is a custom message!

Change Properties

A Simple User Control w/ Properties

Message text changed!

Change Properties

Serwisy

- zawarte w pliku z rozszerzeniem .asmx
- eksportowane metody mają atrybut [WebMethod]
- umożliwiają wygenerowanie dokumentu WSDL dla klienta
- klient może wygenerować klasę opakowującą np. przy pomocy narzędzia WSDL.exe

```
<%@ WebService Language="C#" Class="MathService" %>

using System;
using System.Web.Services;

public class MathService : WebService {

    [WebMethod]
    public float Add(float a, float b)
    {
        return a + b;
    }

    [WebMethod]
    public float Subtract(float a, float b)
    {
        return a - b;
    }

    [WebMethod]
    public float Multiply(float a, float b)
    {
        return a * b;
    }

    [WebMethod]
    public float Divide(float a, float b)
    {
        if (b==0) return -1;
        return a / b;
    }

}
```

MathService

The following operations are supported. For a formal definition, please review the Service Description.

- Multiply
- Divide
- Add
- Subtract

This web service is using http://tempuri.org/ as its default namespace.

Recommendation: Change the default namespace before the XML Web service is made public.

Each XML Web service needs a unique namespace in order for client applications to distinguish it from other services on the Web. http://tempuri.org/ is available for XML Web services that are under development, but published XML Web services should use a more permanent namespace.

Your XML Web service should be identified by a namespace that you control. For example, you can use your company's Internet domain name as part of the namespace. Although many XML Web service namespaces look like URLs, they need not point to actual resources on the Web. (XML Web service namespaces are URIs.)

For XML Web services creating using ASP.NET, the default namespace can be changed using the WebService attribute's Namespace property. The WebService attribute is an attribute applied to the class that contains the XML Web service methods. Below is a code example that sets the namespace to "http://microsoft.com/webservices/":

C#

```
[WebService(Namespace="http://microsoft.com/webservices/")]
public class MyWebService {
    // implementation
}
```

Visual Basic.NET

```
<WebService(Namespace:="http://microsoft.com/webservices/")> Public Class MyWebService
    ' implementation
End Class
```

For more details on XML namespaces, see the W3C recommendation on Namespaces in XML.

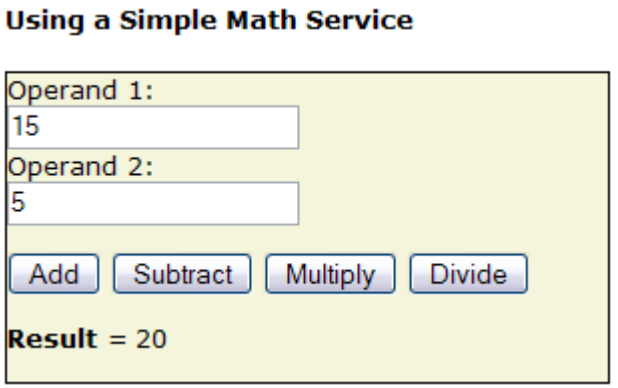
For more details on WSDL, see the WSDL Specification.

For more details on URIs, see RFC 2396.

```
//------------------------------------------------------------------------------
// <autogenerated>
//     This code was generated by a tool.
//     Runtime Version: 1.0.3705.0
//
//     Changes to this file may cause incorrect behavior and will be lost if
//     the code is regenerated.
// </autogenerated>
//------------------------------------------------------------------------------

//
// This source code was auto-generated by wsdl, Version=1.0.3705.0.
//
namespace MathService {
    using System.Diagnostics;
    using System.Xml.Serialization;
    using System;
    using System.Web.Services.Protocols;
    using System.ComponentModel;
    using System.Web.Services;


    /// <remarks/>
    [System.Diagnostics.DebuggerStepThroughAttribute()]
    [System.ComponentModel.DesignerCategoryAttribute("code")]
    [System.Web.Services.WebServiceBindingAttribute(Name="MathServiceSoap",
    Namespace="http://tempuri.org/")]
    public class MathService : System.Web.Services.Protocols.SoapHttpClientProtocol {

        /// <remarks/>
        public MathService() {
            this.Url = "http://localhost/quickstart/aspplus/samples/services/MathService/CS
             /MathService.a" + "smx";
        }

        /// <remarks/>
        [System.Web.Services.Protocols.SoapDocumentMethodAttribute("http://tempuri.org/Add",
        RequestNamespace="http://tempuri.org/", ResponseNamespace="http://tempuri.org/",
        Use=System.Web.Services.Description.SoapBindingUse.Literal,
        ParameterStyle=System.Web.Services.Protocols.SoapParameterStyle.Wrapped)]
        public System.Single Add(System.Single a, System.Single b) {
            object[] results = this.Invoke("Add", new object[] {
                        a,
                        b});
            return ((System.Single)(results[0]));
        }

        /// <remarks/>
        public System.IAsyncResult BeginAdd(System.Single a, System.Single b,
        System.AsyncCallback callback, object asyncState) {
            return this.BeginInvoke("Add", new object[] {
                        a,
                        b}, callback, asyncState);
        }

        /// <remarks/>
        public System.Single EndAdd(System.IAsyncResult asyncResult) {
            object[] results = this.EndInvoke(asyncResult);
            return ((System.Single)(results[0]));
        }
```

```
<%@ Import Namespace="MathService" %>

<html>

<script language="C#" runat="server">

  float operand1 = 0;
  float operand2 = 0;

  public void Submit_Click(Object sender, EventArgs E)
  {
      try
      {
        operand1 = float.Parse(Operand1.Text);
        operand2 = float.Parse(Operand2.Text);
      }
      catch (Exception) { /* ignored */ }

      MathService service = new MathService();

        switch (((Control)sender).ID)
        {
          case "Add"      : Result.Text = "<b>Result</b> = " +
             service.Add(operand1, operand2).ToString(); break;
          case "Subtract" : Result.Text = "<b>Result</b> = " +
             service.Subtract(operand1, operand2).ToString(); break;
          case "Multiply" : Result.Text = "<b>Result</b> = " +
             service.Multiply(operand1, operand2).ToString(); break;
          case "Divide"   : Result.Text = "<b>Result</b> = " +
             service.Divide(operand1, operand2).ToString(); break;
        }
  }

</script>
```

```
<body style="font: 10pt verdana">

  <h4>Using a Simple Math Service </h4>

  <form runat="server">
  <div style="padding:15,15,15,15;background-color:beige;width:300;
   border-color:black;border-width:1;border-style:solid">

    Operand 1: <br><asp:TextBox id="Operand1" Text="15"
       runat="server"/><br>
    Operand 2: <br><asp:TextBox id="Operand2" Text="5"
       runat="server"/><p>

    <input type="submit" id="Add" value="Add"
       OnServerClick="Submit_Click" runat="server">
    <input type="submit" id="Subtract" value="Subtract"
       OnServerClick="Submit_Click" runat="server">
    <input type="submit" id="Multiply" value="Multiply"
       OnServerClick="Submit_Click" runat="server">
    <input type="submit" id="Divide" value="Divide"
       OnServerClick="Submit_Click" runat="server">
    <p>
    <asp:Label id="Result" runat="server"/>
  </div>

  </form>

</body>
</html>
```



aplikacje ASP.NET

- wszystko co zawarte jest w wirtualnym katalogu (i jego podkatalogach) na serwerze
- opcjonalny plik Global.aspx definiujący rozszerzenia klasy HttpApplication
- możliwość zawarcia kodu w metodach `Application_Start`, `Application_End`, `Session_Start`, `Application_Error`, `Session_End`
- możliwość przechowywania obiektów we własności Application (dostępnej również z poziomu strony)

Global.aspx

```
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.IO" %>

<script language="C#" runat="server">

    void Application_Start(Object sender, EventArgs e) {

        DataSet ds = new DataSet();

        FileStream fs = new FileStream(
          Server.MapPath("schemadata.xml"),
          FileMode.Open,FileAccess.Read);
        StreamReader reader = new StreamReader(fs);
        ds.ReadXml(reader);
        fs.Close();

        DataView view = new DataView(ds.Tables[0]);
        Application["Source"] = view;
    }

</script>
```

Application2.aspx

```
<%@ Import Namespace="System.Data" %>

<html>

  <script language="C#" runat="server">

      void Page_Load(Object Src, EventArgs E ) {

          DataView Source = (DataView)(Application["Source"]);
          MySpan.Controls.Add(new LiteralControl
             (Source.Table.TableName));
          MyDataGrid.DataSource = Source;
          MyDataGrid.DataBind();
      }

  </script>

<body>
  <h3><font face="Verdana">Reading Data in Application_OnStart
   </font></h3>
  <h4><font face="Verdana">XML Data for Table: <asp:PlaceHolder
   runat="server" id="MySpan"/></font></h4>

  <ASP:DataGrid id="MyDataGrid" runat="server"
    Width="900"
    BackColor="#ccccff"
    BorderColor="black"
    ShowFooter="false"
    CellPadding=3
    CellSpacing="0"
    Font-Name="Verdana"
    Font-Size="8pt"
    HeaderStyle-BackColor="#aaaadd"
    EnableViewState="false"
  />
</body>
</html>
```

Reading Data in Application_OnStart

XML Data for Table: Table

ProductID	CategoryID	ProductName	ProductDescription	UnitPrice	ImagePath	ServingSize	Servings	Quantity	MinOnHand	MaxOnHand	Manufacturer
1001	1	Chocolate City Milk	Chocolate City Milk Description	2	/quickstart/aspplus/images/milk5.gif	8 fl oz (240 mL)	8	0	0	0	Chocolate City
1002	1	Bessie Brand 2% Milk	Bessie Brand 2% Milk Description	1.19	/quickstart/aspplus/images/milk1.gif	8 fl oz (240 mL)	8	0	0	0	Milk Factory
1003	1	Funny Farms Milk	Funny Farms Whole Milk Description	1.29	/quickstart/aspplus/images/milk4.gif	8 fl oz (240 mL)	10	0	0	0	Funny Farms
2001	2	Fruity Pops	Fruity Pops Description	4.07	/quickstart/aspplus/images/cereal7.gif	3/4 cup (30 g)	17	0	0	0	River Mills
2002	2	U.F.O.s Cereal	U.F.O.s Cereal Description	3.34	/quickstart/aspplus/images/cereal3.gif	1 cup (30 g)	10	0	0	0	Acme Harvesters
2003	2	Healthy Grains	Healthy Grains Cereal Description	3.78	/quickstart/aspplus/images/cereal1.gif	3/4 cup (30 g)	17	0	0	0	All Natural Co.
2004	2	Super Sugar Strike	Super Sugar Strike Description	4.17	/quickstart/aspplus/images/cereal6.gif	3/4 cup (30 g)	17	0	0	0	Capitol Cereals
3001	3	Purple Rain	Brown Barrel Root Beer Description	1.1	/quickstart/aspplus/images/soda5.gif	4 fl oz (120 mL)	8	0	0	0	BrainFade, Inc.
3002	3	Extreme Orange	Bargain Cola Description	0.89	/quickstart/aspplus/images/soda6.gif	6 fl oz (180 mL)	6	0	0	0	SuperX Beverages
3003	3	Kona Diet Cola	Super Red Pop Soda Description	1.1	/quickstart/aspplus/images/soda7.gif	4 fl oz (120 mL)	10	0	0	0	Kona Kola Co.
3004	3	Fizzy Fizzing Drink	Lemon Lime Quencher Description	1.05	/quickstart/aspplus/images/soda8.gif	6 fl oz (180 mL)	5	0	0	0	Sparkle Co.
1005	1	Marigold Whole Milk	Marigold Whole Milk Description	1.39	/quickstart/aspplus/images/milk6.gif	8 fl oz (240 mL)	8	0	0	0	Marigold Meadows

Stan aplikacji - sesja

Global.asax

```
<script language="C#" runat="server">

    void Session_Start(Object sender, EventArgs e) {
      Session["BackColor"] = "beige";
      Session["ForeColor"] = "black";
      Session["LinkColor"] = "blue";
      Session["FontSize"] = "8pt";
      Session["FontName"] = "verdana";
    }
</script>
```

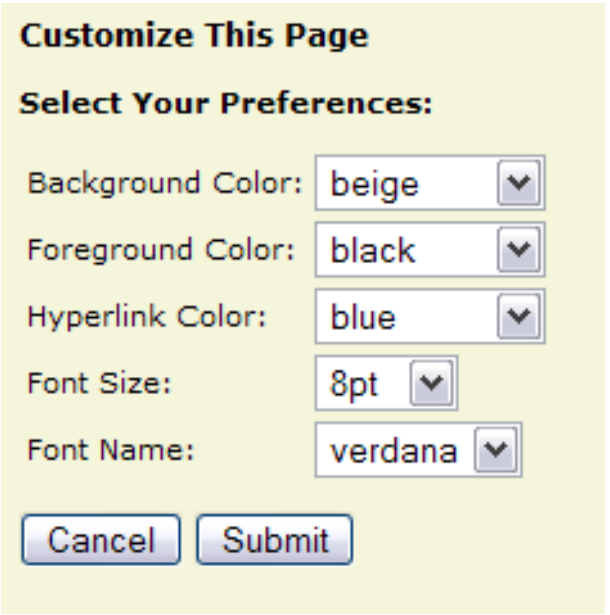
Session1.aspx

```
<html>
  <script language="C#" runat="server">
      String GetStyle(String key) {
        return Session[key].ToString();
      }
  </script>

  <style>
    body {
      font: <%=GetStyle("FontSize")%> <%=GetStyle("FontName")%>;
      background-color: <%=GetStyle("BackColor")%>;
    }
    a { color: <%=GetStyle("LinkColor")%> }
  </style>
  <body style="color:<%=GetStyle("ForeColor")%>">
    <h3><font face="Verdana">Storing Volatile Data in Session State
       </font></h3>
    <b><a href="customize.aspx">Customize This Page</a></b><p>
    Imagine some content here ...<br>
  </body>
</html>
```

customize.aspx

```
<html>

  <script language="C#" runat="server">

      void Page_Load(Object sender, EventArgs E) {

          if (!Page.IsPostBack) {
            ViewState["Referer"] = Request.Headers["Referer"];
            BackColor.Value = (String)Session["BackColor"];
            ForeColor.Value = (String)Session["ForeColor"];
            LinkColor.Value = (String)Session["LinkColor"];
            FontSize.Value = (String)Session["FontSize"];
            FontName.Value = (String)Session["FontName"];
          }
      }

      void Submit_Click(Object sender, EventArgs E) {
        Session["BackColor"] = BackColor.Value;
        Session["ForeColor"] = ForeColor.Value;
        Session["LinkColor"] = LinkColor.Value;
        Session["FontSize"]  = FontSize.Value;
        Session["FontName"]  = FontName.Value;

        if ( ViewState["Referer"] != null ){
          Response.Redirect(ViewState["Referer"].ToString());
        }
      }

      void Cancel_Click(Object sender, EventArgs E) {
        if ( ViewState["Referer"] != null ){
          Response.Redirect(ViewState["Referer"].ToString());
        }
      }

      String GetStyle(String key) {
        return Session[key].ToString();
      }
  </script>
```

Stan aplikacji - sesja



Stan aplikacji - cookies

Cookies1.aspx

```
<html>
  <script language="C#" runat="server">
      void Page_Load(Object sender, EventArgs E) {
        if (Request.Cookies["preferences1"] == null) {
          HttpCookie cookie = new HttpCookie("preferences1");
          cookie.Values.Add("ForeColor","black");
          cookie.Values.Add("BackColor","beige");
          cookie.Values.Add("LinkColor","blue");
          cookie.Values.Add("FontSize","8pt");
          cookie.Values.Add("FontName","Verdana");
          Response.AppendCookie(cookie);
        }
      }
      protected String GetStyle(String key) {
        HttpCookie cookie = Request.Cookies["preferences1"];
        if (cookie != null) {
          switch (key) {
            case "ForeColor" :
              return cookie.Values["ForeColor"]; break;
            case "BackColor" :
              return cookie.Values["BackColor"]; break;
            case "LinkColor" :
              return cookie.Values["LinkColor"]; break;
            case "FontSize" :
              return cookie.Values["FontSize"]; break;
            case "FontName" :
              return cookie.Values["FontName"]; break;
          }
        }
        return "";
      }
  </script>
```

```
  <style>

    body {
      font: <%=GetStyle("FontSize")%> <%=GetStyle("FontName")%>;
      background-color: <%=GetStyle("BackColor")%>;
    }

    a { color: <%=GetStyle("LinkColor")%> }

  </style>

  <body style="color:<%=GetStyle("ForeColor")%>">

    <h3><font face="Verdana">Storing Volatile Data with
       Client-Side Cookies</font></h3>

    <b><a href="customize.aspx">Customize This Page</a></b><p>

    Imagine some content here ...<br>

  </body>

</html>
```

Customize.aspx

```
<html>
  <script language="C#" runat="server">
      void Page_Load(Object sender, EventArgs E) {
        if (!IsPostBack){
          HttpCookie cookie = Request.Cookies["preferences1"];
          ViewState["Referer"] = Request.Headers["Referer"];
          if ( cookie != null ){
            BackColor.Value = (String)cookie.Values["BackColor"];
            ForeColor.Value = (String)cookie.Values["ForeColor"];
            LinkColor.Value = (String)cookie.Values["LinkColor"];
            FontSize.Value = (String)cookie.Values["FontSize"];
            FontName.Value = (String)cookie.Values["FontName"];
          }
        }
      }
      void Submit_Click(Object sender, EventArgs E) {
        HttpCookie cookie = new HttpCookie("preferences1");
        cookie.Values.Add("ForeColor",ForeColor.Value);
        cookie.Values.Add("BackColor",BackColor.Value);
        cookie.Values.Add("LinkColor",LinkColor.Value);
        cookie.Values.Add("FontSize",FontSize.Value);
        cookie.Values.Add("FontName",FontName.Value);
        Response.AppendCookie(cookie);
        if ( ViewState["Referer"] != null ){
          Response.Redirect(ViewState["Referer"].ToString());
        }
      }
      void Cancel_Click(Object sender, EventArgs E) {
        if ( ViewState["Referer"] != null ){
          Response.Redirect(ViewState["Referer"].ToString());
        }
      }
```

klasa Page

- System.Object
  - System.Web.UI.Control
    - System.Web.UI.TemplateControl
      - System.Web.UI.Page
- `public class Page : TemplateControl, IHttpHandler`
- związana z plikami .aspx
- możliwość dostępu do własności takich jak
  - Application
  - ErrorPage
  - IsPostBack
  - IsValid
  - Request
  - Response
  - Session

klasy HttpRequest, HttpResponse

- System.Object
  - System.Web.HttpRequest
- `public sealed class HttpRequest`
- System.Object
  - System.Web.HttpResponse
- `public sealed class HttpResponse`
- Własności:
  - `public TextWriter Output {get;}`
  - `public Stream OutputStream {get;}`
  - `public string ContentType {get; set;}`
  - `public Encoding ContentEncoding {get; set;}`
- Metody:
  - `public void Redirect(string url);`
  - `public void End();`
  - `public void WriteFile(string filename);`

```
private void Page_Load(object sender, System.EventArgs e)
    {
            //Set the appropriate ContentType.
        Response.ContentType = "Application/pdf";
            //Get the physical path to the file.
        string FilePath = MapPath("acrobat.pdf");
            //Write the file directly to the HTTP content output stream.
        Response.WriteFile(FilePath);
        Response.End();
    }
```

Konfiguracja

- Pliki konfiguracyjne są umieszczone razem z aplikacją
- Pliki konfiguracyjne są tekstowe (XML)
- Zmiany w konfiguracji są automatycznie wykrywane, nie ma konieczności restartów
- Plik konfiguracyjny ma nazwę `web.config` i konfiguruje katalog w którym jest umieszczony (wyjątkiem jest plik konfiguracyjny komputera: `machine.config`)
- Konfiguracje są dziedziczone w podkatalogach

Własne parametry konfiguracyjne

- Zawarte w sekcji `appSettings`
- Parametry są zapisywane jako pary nazwa-wartość
- Można nimi manipulować przy pomocy klasy `ConfigurationSettings`:
  - statyczna właściwość `AppSettings`
  - udostępnia kolekcję `NameValueCollection`, która zawiera pary łańcuchów (klucz-wartość) i obsługuje zarówno iteratory, jak i indeksowanie po kluczu

```
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.SqlClient" %>
<%@ Import Namespace="System.Configuration" %>

<html>

<script language="C#" runat="server">

    void Page_Load(Object Src, EventArgs E ) {

        String dsn = ConfigurationSettings.AppSettings["pubs"];

        SqlConnection myConnection = new SqlConnection(dsn);
        SqlDataAdapter myCommand = new SqlDataAdapter
          ("select * from Authors", myConnection);

        DataSet ds = new DataSet();
        myCommand.Fill(ds, "Authors");

        MyDataGrid.DataSource=new DataView(ds.Tables[0]);
        MyDataGrid.DataBind();
    }

</script>

<body>


</body>
</html>
```

web.config

```
<configuration>
  <appSettings>
    <add key="pubs" value="server=(local)\NetSDK;database=pubs;
       Integrated Security=SSPI" />
  </appSettings>
</configuration>
```

Serializacja

- Zapisywanie obiektów do pamięci nieulotnej
- Prekazywanie obiektów pomiędzy aplikacjami
- Automatyczne zapisywanie i odtwarzanie obiektów, drzew obiektów, obsługa pętli odwołań
- Wymagany jest atrybut Serializable (nie jest dziedziczony)
- Domyślnie zapisywane są wszystkie pola, również prywatne
- Można wskazać pola które nie są zapisywane
- Można zastąpić domyślną procedurę serializacji implementując interfejs ISerializable
- Można serializować do formatu binarnego lub SOAP

```
[Serializable]
public class MyObject {
  public int n1 = 0;
  public int n2 = 0;
  public String str = null;
}


MyObject obj = new MyObject();
obj.n1 = 1;
obj.n2 = 24;
obj.str = "Some String";
IFormatter formatter = new BinaryFormatter();
Stream stream = new FileStream("MyFile.bin", FileMode.Create,
   FileAccess.Write, FileShare.None);
formatter.Serialize(stream, obj);
stream.Close();


IFormatter formatter = new BinaryFormatter();
Stream stream = new FileStream("MyFile.bin", FileMode.Open,
   FileAccess.Read, FileShare.Read);
MyObject obj = (MyObject) formatter.Deserialize(stream);
stream.Close();



[Serializable]
public class MyObject
{
  public int n1;
  [NonSerialized] public int n2;
  public String str;
}
```

```
[Serializable]
public class MyObject : ISerializable
{
  public int n1;
  public int n2;
  public String str;

  public MyObject()
  {
  }

  protected MyObject(SerializationInfo info, StreamingContext context)
  {
    n1 = info.GetInt32("i");
    n2 = info.GetInt32("j");
    str = info.GetString("k");
  }
  [SecurityPermissionAttribute(SecurityAction.Demand,
   SerializationFormatter=true)]
  public virtual void GetObjectData(SerializationInfo info,
    StreamingContext context)
  {
    info.AddValue("i", n1);
    info.AddValue("j", n2);
    info.AddValue("k", str);
  }
}
```

```
[Serializable]
public class ObjectTwo : MyObject
{
  public int num;

  public ObjectTwo() : base()
  {
  }

  protected ObjectTwo(SerializationInfo si, StreamingContext context)
    : base(si,context)
  {
    num = si.GetInt32("num");
  }
[SecurityPermissionAttribute(SecurityAction.Demand,
SerializationFormatter=true)]
  public override void GetObjectData(SerializationInfo si,
   StreamingContext context)
  {
    base.GetObjectData(si,context);
    si.AddValue("num", num);
  }
}
```

```
using System;
using System.IO;
using System.Collections;
using System.Runtime.Serialization.Formatters.Binary;
using System.Runtime.Serialization;

[Serializable]
public sealed class Singleton : ISerializable
{
    private static readonly Singleton theOneObject = new Singleton();

    public String someString;
    public Int32 someNumber;

    private Singleton()
    {
        someString = "This is a string field";
        someNumber = 123;
    }

    public static Singleton GetSingleton()
    {
        return theOneObject;
    }

    void ISerializable.GetObjectData(
        SerializationInfo info, StreamingContext context)
    {
        info.SetType(typeof(SingletonSerializationHelper));
    }

}
[Serializable]
internal sealed class SingletonSerializationHelper : IObjectReference
{
    // GetRealObject is called after this object is deserialized.
    public Object GetRealObject(StreamingContext context)
    {
        return Singleton.GetSingleton();
    }
}
```

```
class App
{
    [STAThread]
    static void Main()
    {
        FileStream fs = new FileStream("DataFile.dat", FileMode.Create);
        try
        {
            BinaryFormatter formatter = new BinaryFormatter();
            Singleton[] a1 = { Singleton.GetSingleton(),
                Singleton.GetSingleton() };
            // This displays "True".
            Console.WriteLine(
                "Do both array elements refer to the same object? " +
                (a1[0] == a1[1]));
            formatter.Serialize(fs, a1);
            fs.Position = 0;
            Singleton[] a2 = (Singleton[]) formatter.Deserialize(fs);
            // This displays "True".
            Console.WriteLine("Do both array elements refer to the same
                object? "
                + (a2[0] == a2[1]));
            // This displays "True".
            Console.WriteLine("Do all array elements refer to the same
                object? "
                + (a1[0] == a2[0]));
        }
        catch (SerializationException e)
        {
            Console.WriteLine("Failed to serialize. Reason: " +
                e.Message);
            throw;
        }
        finally
        {
            fs.Close();
        }
    }
}
```

strony Master

- Umożliwiają utrzymanie jednolitego stylu witryny
- Mogą zawierać tekst, elementy HTML, kontrolki serwerowe
- Mogą być zagnieżdżane
- Mają rozszerzenie .master i dyrektywę @Master (zamiast @Page)

strony Master

- rejony gdzie pojawi się (zmienna) zawartość są określone przez kontrolki ContentPlaceHolder

```
<%@ Master Language="C#" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
    1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html  >
<head runat="server" >
    <title>Master page title</title>
</head>
<body>
    <form id="form1" runat="server">
        <table>
            <tr>
                <td><asp:contentplaceholder id="Main" runat="server" />
                    </td>
                <td><asp:contentplaceholder id="Footer" runat="server" />
                    </td>
            </tr>
        </table>
    </form>
</body>
</html>
```

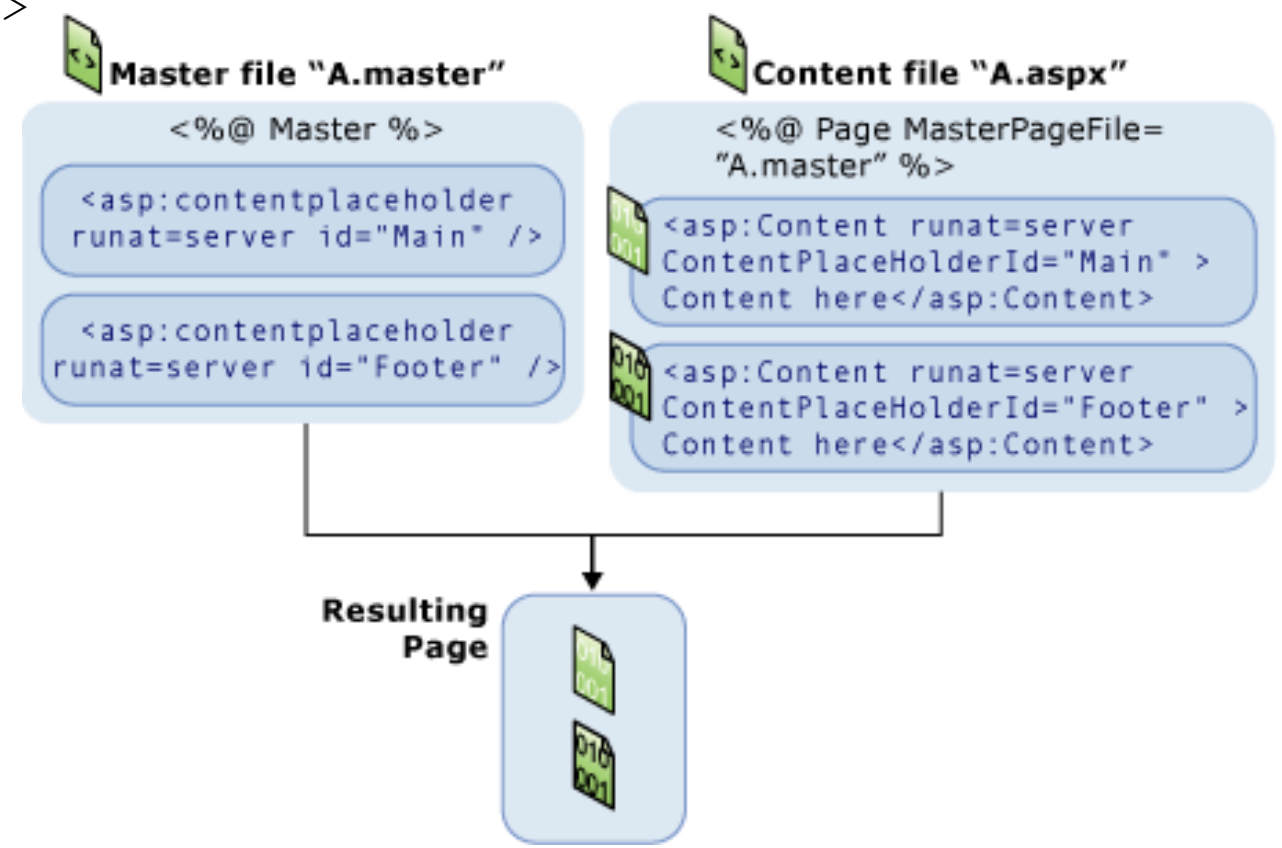
strony Master - treść

- strony z treścią są oparte o typowe strony z rozszerzeniem .aspx
- są dołączone do strony master za pomocą atrybutu MasterPageFile
- na stronach z treścią umieszcz się kontrolki Content i łączy z ContentPlaceHolder

```
<%@ Page Language="C#" MasterPageFile="~/Master1.master" Title="Content Page"%>
```

strony Master - łączenie

```
<% @ Page Language="C#" MasterPageFile="~/Master.master"
    Title="Content Page 1" %>
<asp:Content ID="Content1" ContentPlaceHolderID="Main" Runat="Server">
    Main content.
</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="Footer" Runat="Server">
    Footer content.
</asp:content>
```



mapa witryny

- Umożliwia opisanie struktury witryny (zależności pomiędzy stronami wchodzącymi w jej skład)
- Z opisu korzystają specjalne kontrolki
- Opis jest (typowo) plikiem XML, wykorzystującym elementy siteMap i siteMapNode
- Musi być tylko jeden element siteMapNode najwyższego poziomu
- Domyślna nazwa pliku: Web.sitemap

mapa witryny

```
<siteMap>
  <siteMapNode title="Home" description="Home" url="~/default.aspx">
    <siteMapNode title="Products" description="Our products"
      url="~/Products.aspx">
      <siteMapNode title="Hardware" description="Hardware choices"
        url="~/Hardware.aspx" />
      <siteMapNode title="Software" description="Software choices"
        url="~/Software.aspx" />
    </siteMapNode>
    <siteMapNode title="Services" description="Services we offer"
        url="~/Services.aspx">
        <siteMapNode title="Training" description="Training classes"
          url="~/Training.aspx" />
        <siteMapNode title="Consulting" description="Consulting services"
          url="~/Consulting.aspx" />
        <siteMapNode title="Support" description="Supports plans"
          url="~/Support.aspx" />
    </siteMapNode>
  </siteMapNode>
</siteMap>
```

Wykorzystanie mapy

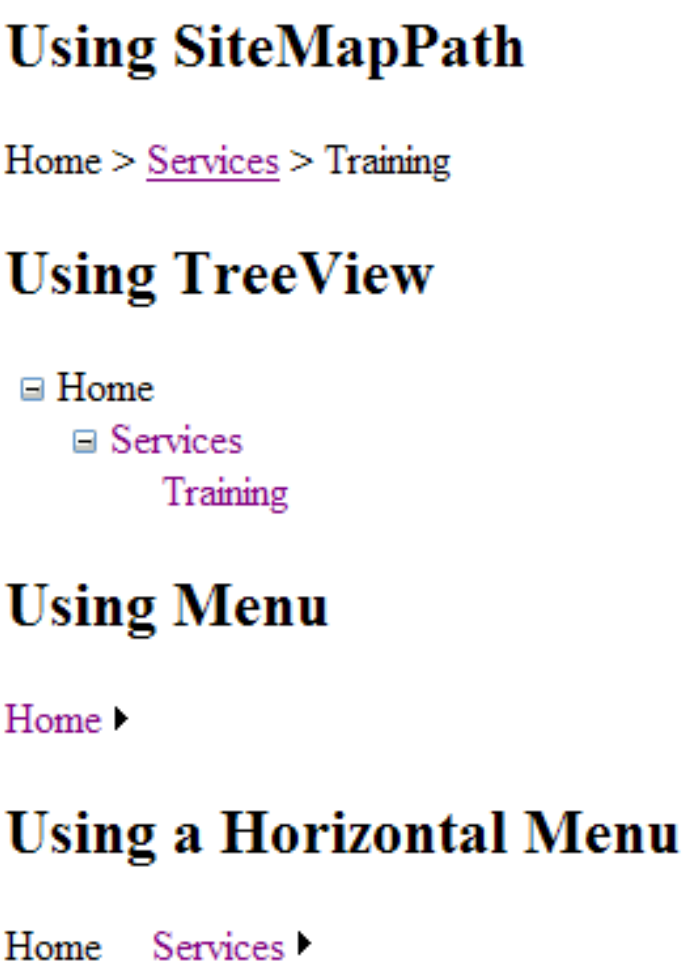
- Specjalizowane kontrolki korzystają z mapy i w różny sposób wyświetlają strukturę witryny:
  - SiteMapPath
  - TreeView
  - Menu

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap>
  <siteMapNode title="Home" >
    <siteMapNode title="Services" >
      <siteMapNode title="Training" url="~/Training.aspx"/>
    </siteMapNode>
  </siteMapNode>
</siteMap>
```

wykorzystanie mapy

```
<form id="form1" runat="server">
<div>
<h2>Using SiteMapPath</h2>
<asp:SiteMapPath ID="SiteMapPath1" Runat="server">
</asp:SiteMapPath>
<asp:SiteMapDataSource ID="SiteMapDataSource1" Runat="server" />
<h2>Using TreeView</h2>
<asp:TreeView ID="TreeView1" Runat="Server"
    DataSourceID="SiteMapDataSource1">
</asp:TreeView>
<h2>Using Menu</h2>
<asp:Menu ID="Menu2" Runat="server" DataSourceID="SiteMapDataSource1">
</asp:Menu>
<h2>Using a Horizontal Menu</h2>
<asp:Menu ID="Menu1" Runat="server" DataSourceID="SiteMapDataSource1"
  Orientation="Horizontal"
  StaticDisplayLevels="2" >
</asp:Menu>

</div>
</form>
```

wykorzystanie mapy



Themes (tematy?)

- Nowe w ASP .NET 2.0
- Umożliwiają określenie styli kontrolek w oddzieleniu od kodu samej strony
- Dzięki temu można opracować stronę bez uwzglenienia styli, a potem stosować różne style bez modyfikacji strony
- Można ściągać tematy z zewnętrznych źródeł
- Znajdują się w folderze App_Themes
- Są też globalne tematy, dostępne dla wszystkich
- Pliki mają rozszerzenie .skin

Themes

- Zawartością pliku .skin są definicje kontrolek, prawie identyczne z tymi które można umieścić na zwykłej stronie - różnicą jest brak atrybutu ID
- Własności kontrolek zdefiniowane w pliku .skin nadpisują własności zdefiniowane w pliku strony
- Temat przypisuje się do strony przy użyciu dyrektywy <%@ Page Theme="..." %>. Wskazuje się w ten sposób folder gdzie znajdują się pliki .skin (lokalny dla aplikacji lub globalny)

Themes

- Możliwe jest również ustawienie tematów dla wszystkich stron aplikacji. W tym celu w pliku web.config należy wpisać <pages theme="..."/>
- Pojedyncza strona może być wtedy wyłączona z zastosowania tematu poprzez dyrektywę <%@ Page Theme="" %> (pusty łańcuch)
- Tematu nie można przypisać do strony Master, a jedynie do zwykłej
- Można wyłączyć pojedynczą kontrolkę z zastosowania tematu poprzez ustawienie własności EnableTheming na false

```
<%@ Page Language="C#" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
  <title>Page with No Theme Applied</title>
</head>
<body>
  <form id="form1" runat="server">
    <div>
      <h3>
        Page with No Theme Applied</h3>
      <asp:Label ID="Label1" runat="server" Text="Hello 1" Font-Bold="true" ForeColor="orange" /><br />
      <asp:Label ID="Label2" runat="server" Text="Hello 2" Font-Bold="true" ForeColor="orange" /><br />
      <asp:Label ID="Label3" runat="server" Text="Hello 3" Font-Bold="true" ForeColor="orange" /><br />
      <asp:Calendar BackColor="White" BorderColor="Black" BorderStyle="Solid" CellSpacing="1"
        Font-Names="Verdana" Font-Size="9pt" ForeColor="Black" Height="250px" ID="Calendar1"
        NextPrevFormat="ShortMonth" runat="server" Width="330px">
        <SelectedDayStyle BackColor="#333399" ForeColor="White" />
        <OtherMonthDayStyle ForeColor="#999999" />
        <TodayDayStyle BackColor="#999999" ForeColor="White" />
        <DayStyle BackColor="#CCCCCC" />
        <NextPrevStyle Font-Bold="True" Font-Size="8pt" ForeColor="White" />
        <DayHeaderStyle Font-Bold="True" Font-Size="8pt" ForeColor="#333333" Height="8pt" />
        <TitleStyle BackColor="#333399" BorderStyle="Solid" Font-Bold="True" Font-Size="12pt"
          ForeColor="White" Height="12pt" />
      </asp:Calendar>
      <br />
    </div>
  </form>
</body>
</html>
```

```
<%@ Page Language="C#" Theme="ExampleTheme" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
  <title>Page with Example Theme Applied</title>
</head>
<body>
  <form id="form1" runat="server">
    <div>
      <h3>
        Page with Example Theme Applied</h3>
      <asp:Label ID="Label1" runat="server" Text="Hello 1" /><br />
      <asp:Label ID="Label2" runat="server" Text="Hello 2" /><br />
      <asp:Label ID="Label3" runat="server" Text="Hello 3" /><br />
      <asp:Calendar ID="Calendar1" runat="server"/>
      <br />
    </div>
  </form>
</body>
</html>
```

W folderze ExampleTheme jest poniższy plik .skin

```
<asp:Label Font-Bold="true" ForeColor="orange" runat="server" />

<asp:Calendar BackColor="White" BorderColor="Black" BorderStyle="Solid" CellSpacing="1"
  Font-Names="Verdana" Font-Size="9pt" ForeColor="Black" Height="250px"
  NextPrevFormat="ShortMonth" Width="330px" runat="server">
  <SelectedDayStyle BackColor="#333399" ForeColor="White" />
  <OtherMonthDayStyle ForeColor="#999999" />
  <TodayDayStyle BackColor="#999999" ForeColor="White" />
  <DayStyle BackColor="#CCCCCC" />
  <NextPrevStyle Font-Bold="True" Font-Size="8pt" ForeColor="White" />
  <DayHeaderStyle Font-Bold="True" Font-Size="8pt" ForeColor="#333333" Height="8pt" />
  <TitleStyle BackColor="#333399" BorderStyle="Solid" Font-Bold="True" Font-Size="12pt"
    ForeColor="White" Height="12pt" />
</asp:Calendar>
```

Skórki z nazwą

- Domyślnie definicja stylu kontrolki z pliku .skin jest stosowana do wszystkich kontrolek danego typu na stronie
- Przy użyciu nazwanych skórek można jednak wymusić różny wygląd kontrolek danego typu
- W tym celu definicję w pliku .skin należy opatrzyć własnością SkinID, z przypisaną wybraną wartością. Również w pliku strony należy przypisać SkinID z tą samą wartością do wybranych kontrolek. Pozostałe będą miały zastosowany temat domyslny (bez SkinID)

```
<%@ Page Language="C#" Theme="OrangeTheme2" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
  <title>Named Skins</title>
</head>
<body>
  <form id="form1" runat="server">
    <div>
    <h3>Named Skins</h3>
      <asp:Label ID="Label1" runat="server" Text="Hello 1" /><br />
      <asp:Label ID="Label2" runat="server" Text="Hello 2" SkinID="Blue" /><br />
      <asp:Label ID="Label3" runat="server" Text="Hello 3" /><br />
      <br />
      <asp:Calendar ID="Calendar1" runat="server"/>
      <br />
      <asp:Calendar ID="Calendar2" SkinID="Simple" runat="server"/>
    </div>
  </form>
</body>
</html>
```

W folderze OrangeTheme znajdują się pliki Label.skin i Calendar.skin

Label.skin

```
<asp:label runat="server" font-bold="true" forecolor="orange" />
<asp:label runat="server" SkinID="Blue" font-bold="true" forecolor="blue" />
```

Calendar.skin

```
<asp:Calendar runat="server" BackColor="#FFFFCC" BorderColor="#FFCC66" BorderWidth="1px"
DayNameFormat="FirstLetter" Font-Names="Verdana" Font-Size="8pt" ForeColor="#663399"
Height="200px" ShowGridLines="True" Width="220px">
  <SelectedDayStyle BackColor="#CCCCFF" Font-Bold="True" />
  <SelectorStyle BackColor="#FFCC66" />
  <OtherMonthDayStyle ForeColor="#CC9966" />
  <TodayDayStyle BackColor="#FFCC66" ForeColor="White" />
  <NextPrevStyle Font-Size="9pt" ForeColor="#FFFFCC" />
  <DayHeaderStyle BackColor="#FFCC66" Font-Bold="True" Height="1px" />
  <TitleStyle BackColor="#990000" Font-Bold="True" Font-Size="9pt" ForeColor="#FFFFCC" />
</asp:Calendar>

<asp:Calendar SkinID="Simple" runat="server" BackColor="White" BorderColor="#999999"
CellPadding="4" DayNameFormat="FirstLetter" Font-Names="Verdana" Font-Size="8pt"
ForeColor="Black" Height="180px" Width="200px">
  <SelectedDayStyle BackColor="#666666" Font-Bold="True" ForeColor="White" />
  <SelectorStyle BackColor="#CCCCCC" />
  <WeekendDayStyle BackColor="#FFFFCC" />
  <OtherMonthDayStyle ForeColor="#808080" />
  <TodayDayStyle BackColor="#CCCCCC" ForeColor="Black" />
  <NextPrevStyle VerticalAlign="Bottom" />
  <DayHeaderStyle BackColor="#CCCCCC" Font-Bold="True" Font-Size="7pt" />
  <TitleStyle BackColor="#999999" BorderColor="Black" Font-Bold="True" />
</asp:Calendar>
```

Named Skins

Hello 1
Hello 2
Hello 3

≤	January 2007					≥
S	M	T	W	T	F	S
31	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	9	10

≤	January 2007					≥
S	M	T	W	T	F	S
31	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	9	10

Style serwerowe przy użyciu tematów

- Możliwe jest zastosowanie tematów tak, że będą one określały domyślny styl kontrolki, który może zostać zmieniony przez zapis w samej kontrolce (normalnie jest odwrotnie)
- Należy ustawić atrybut StyleSheetTheme dyrektywy @Page (lub sekcji <pages/>)
- Jeśli stosowany jest zarówno StyleSheetTheme, jak i normalny Theme, kolejność stosowania styli jest następująca:
  - StyleSheetTheme
  - styl wpisany w kontrolkę na stronie
  - Theme

Normalny Theme:

```
<%@ Page Language="C#" Theme="OrangeTheme" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
  <title>Theme Overrides Page Properties</title>
</head>
<body>
  <form id="form1" runat="server">
    <div>
    <h3>Properties in the Page are Overridden By Theme</h3>
      <asp:Label ID="Label1" runat="server" Text="Hello 1" /><br />
      <asp:Label ID="Label2" runat="server" Text="Hello 2" ForeColor="blue" />
      &lt;-- Notice that this Label is orange (from Theme) instead of blue (from Page)
      <br />
      <asp:Label ID="Label3" runat="server" Text="Hello 3" /><br />
    </div>
  </form>
</body>
</html>
```

W katalogu OrangeTheme:

```
<asp:label runat="server" font-bold="true" forecolor="orange" />
```

StyleSheetTheme:

```
<%@ Page Language="C#" StyleSheetTheme="OrangeTheme" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
  <title>Page Properties Override StyleSheetTheme</title>
</head>
<body>
  <form id="form1" runat="server">
    <div>
    <h3>Properties in the Page Override StyleSheetTheme</h3>
      <asp:Label ID="Label1" runat="server" Text="Hello 1" /><br />
      <asp:Label ID="Label2" runat="server" Text="Hello 2" ForeColor="blue" />
      &lt;-- Notice that this Label is blue (from Page) instead of orange (from StyleSheetTheme)
      <br />
      <asp:Label ID="Label3" runat="server" Text="Hello 3" /><br />
    </div>
  </form>
</body>
</html>
```

W katalogu OrangeTheme tak jak poprzednio:

```
<asp:label runat="server" font-bold="true" forecolor="orange" />
```

Properties in the Page are Overridden By Theme

Hello 1
Hello 2 <-- Notice that this Label is orange (from Theme) instead of blue (from Page)
Hello 3

Properties in the Page Override StyleSheetTheme

Hello 1
Hello 2 <-- Notice that this Label is blue (from Page) instead of orange (from StyleSheetTheme)
Hello 3

Logowanie

- ASP .NET 2.0 dostarcza różnych możliwości związanych z kontrolą dostępu
- Między innymi można stosować kontrolę z poziomu strony (w przeciwieństwie do kontroli z poziomu systemu Windows)
- Jest to tzw. Forms-based autentication
- Należy aplikację skonfigurować (w web.config) aby używała tego rodzaju kontroli dostępu
- Konfiguracja umożliwia ustalenie takich atrybutów jak użycie cookies, url do którego należy przejść po zalogowaniu, url strony logującej, użycie SSL, timeout etc.

Logowanie

Przykładowa sekcja ustawiająca form-based authentication i uniemożliwiająca dostęp użytkownikom anonimowym

```
<configuration>
  <system.web>
    <authentication mode="Forms"/>
    <authorization>
        <deny users="?" />
    </authorization>
  </system.web>
</configuration>
```

Sekcja z zaawansowanymi ustawieniami

```
<authentication mode="Forms">
   <forms name=".ASPXCOOKIEDEMO" loginUrl="login.aspx" defaultUrl="default.aspx"
         protection="All" timeout="30" path="/" requireSSL="false"
         slidingExpiration="true" enableCrossAppRedirects="false"
         cookieless="UseDeviceProfile" domain="">
         <!-- protection="[All|None|Encryption|Validation]" -->
         <!-- cookieless="[UseUri | UseCookies | AutoDetect | UseDeviceProfile]" -->
   </forms>
</authentication>
```

login.aspx

```
<%@ Import Namespace="System.Web.Security " %>
<html>
  <script language="C#" runat=server>
    void Login_Click(Object sender, EventArgs E) {
      if ((UserEmail.Value == "someone@www.contoso.com") && (UserPass.Value == "password")) {
        FormsAuthentication.RedirectFromLoginPage(UserEmail.Value, PersistCookie.Checked);
      }
      else {
        Msg.Text = "Invalid Credentials: Please try again";
      }
    }
  </script>
  <body>
    <form runat=server>
      <h3><font face="Verdana">Login Page</font></h3>
      <table>
        <tr>
          <td>Email:</td>
          <td><input id="UserEmail" type="text" runat=server/></td>
          <td><ASP:RequiredFieldValidator ControlToValidate="UserEmail" Display="Static" ErrorMessage="*"
                    runat=server/></td>
        </tr>
        <tr>
          <td>Password:</td>
          <td><input id="UserPass" type=password runat=server/></td>
          <td><ASP:RequiredFieldValidator ControlToValidate="UserPass" Display="Static" ErrorMessage="*"
                    runat=server/></td>
        </tr>
        <tr>
          <td>Persistent Cookie:</td>
          <td><ASP:CheckBox id=PersistCookie runat="server" /> </td>
          <td></td>
        </tr>
      </table>

      <asp:button text="Login" OnClick="Login_Click" runat=server/>
      <p>
      <asp:Label id="Msg" ForeColor="red" Font-Names="Verdana" Font-Size="10" runat=server />
    </form>
  </body>

</html>
```

default.aspx

```
<%@ Import Namespace="System.Web.Security " %>
<html>
  <script language="C#" runat=server>
    void Page_Load(Object Src, EventArgs E ) {
      Welcome.Text = "Hello, " + User.Identity.Name;
    }
    void Signout_Click(Object sender, EventArgs E) {
      FormsAuthentication.SignOut();
      Response.Redirect("login.aspx");
    }
  </script>

  <body>
    <h3><font face="Verdana">Using Cookie Authentication</font></h3>
    <form runat=server>
      <h3><asp:label id="Welcome" runat=server/></h3>
      <asp:button text="Signout" OnClick="Signout_Click" runat=server/>
    </form>
  </body>
</html>
```

web.config

```
<configuration>
  <system.web>
      <authentication mode="Forms">
        <forms name=".ASPXUSERDEMO" loginUrl="login.aspx" protection="All" timeout="60" />
      </authentication>
      <authorization>
        <deny users="?" />
      </authorization>
    <globalization requestEncoding="UTF-8" responseEncoding="UTF-8" />
  </system.web>
</configuration>
```

Login Page

Email:

Password:

Persistent Cookie:

Login

Using Cookie Authentication

Hello, someone@www.contoso.com

Signout

Kontrolki logowania

- ASP .NET 2.0 zawiera specjalne kontrolki umożliwiające logowanie, zmianę hasła, wyświetlanie treści w zależności od osoby zalogowanej, wyświetlanie statusu zalogowania itp.

Kontrolka Login

- Zawiera zawsze pola: user, password i przycisk login
- Może zawierać takie elementy jak:
  - link do przypominacza hasła
  - checkbox właczajacy pamiętanie wpisów
  - link do pomocy
  - link do rejestracji nowego użytkownika
  - tekst instrukcji
  - tekst błędu
- Używa tzw. Membership Provider do składowania danych użytkowników. Może być to np. MS SQL Server
- Wygląd może być dostosowany do potrzeb

Kontrolka LoginStatus

- W zależności od tego czy użytkownik jest zalogowany czy nie, wyświetla link umożliwiający zalogowanie lub wylogowanie
- Zalogowanie jest wykrywane na podstawie własności IsAuthenticated występującej we własności Request obiektu klasy Page
- Może wyświetlać tekst bądź grafikę

Kontrolka CreateUserWizard

- Stanowi interfejs dla Membership Provider umożliwiający utworzenie nowego użytkownika
- Umożliwia wprowadzenie nazwy i hasła
- Opcjonalnie może umożliwiać wprowadzenie adresu e-mail oraz pytania i odpowiedzi w przypadku zapomnianego hasła
- Umożliwia automatyczną generację hasła

```
<?xml version="1.0" ?>
<configuration>
    <connectionStrings>
   <add name="ASPNETDB" connectionString="Server=(local)\SQLExpress;Integrated
Security=SSPI;Database=aspnetdb"/>
   </connectionStrings>
   <system.web>
     <pages styleSheetTheme="SmokeAndGlass"/>
      <authentication mode="Forms">
         <forms    name=".ASPXAUTH"
                loginUrl="Login.aspx"
                protection="All"
                timeout="30"
                path="/"
                requireSSL="false"
                slidingExpiration="true"
                defaultUrl="Login.aspx"
                cookieless="UseDeviceProfile"
                enableCrossAppRedirects="false"/>
      </authentication>
      <membership defaultProvider="QuickStartMembershipSqlProvider"
userIsOnlineTimeWindow="15">
         <providers>
            <add
               name="QuickStartMembershipSqlProvider"
               type="System.Web.Security.SqlMembershipProvider, System.Web, Version=2.0.0.0,
Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a"
               connectionStringName="ASPNETDB"
               enablePasswordRetrieval="false"
               enablePasswordReset="true"
               requiresQuestionAndAnswer="true"
               applicationName="SecurityQuickStart"
               requiresUniqueEmail="true"
               passwordFormat="Hashed"/>
         </providers>
      </membership>
      <roleManager
                enabled="true"
                cacheRolesInCookie="true"
                defaultProvider="QuickStartRoleManagerSqlProvider"
                cookieName=".ASPXROLES"
                cookiePath="/"
                cookieTimeout="30"
                cookieRequireSSL="false"
                cookieSlidingExpiration="true"
                createPersistentCookie="false"
                cookieProtection="All">
         <providers>
            <add name="QuickStartRoleManagerSqlProvider"
               type="System.Web.Security.SqlRoleProvider, System.Web, Version=2.0.0.0, Culture=neutral,
PublicKeyToken=b03f5f7f11d50a3a"
               connectionStringName="ASPNETDB"
               applicationName="SecurityQuickStart"/>
         </providers>
      </roleManager>
   </system.web>
</configuration>
```

Site.master

```
<%@ Master Language="C#" %>
<html>
  <head runat="server" id="Head1">
    <title>
      Using Site Navigation Controls</title>
  </head>
  <body>
    <form id="form1" runat="server">
      <table style="background-color: blue" cellspacing="0" cellpadding="5" border="0"
                          height="80%">
        <tr height="20px">
          <td style="width: 100px">
            <img id="Img1" src="Images/microsoft.gif"  runat="server"/>
          </td>
          <td style="width: 100px">
            <img id="Img2" src="Images/homeadvisor.gif" runat="server" />
          </td>
          <td style="width: 80px; background-color: white" align="center">
            <asp:loginstatus ID="loginstatus" runat="server"/>
          </td>
        </tr>
        <tr>
          <td style="width: 100px" valign="top">
             </td>
          <td style="background-color: white;padding-left:25;padding-top:15" colspan="2" valign="top">
            <asp:ContentPlaceHolder ID="MainBody" runat="server" />
          </td>
        </tr>
      </table>
    </form>
  </body>
</html>
```

Home.aspx

```
<%@ Page Language="C#" MasterPageFile="~/Site.master"%>

<asp:Content ID="Content1" ContentPlaceHolderId="MainBody" runat="server">
  <h1>Welcome to Login Controls</h1>
</asp:Content>
```

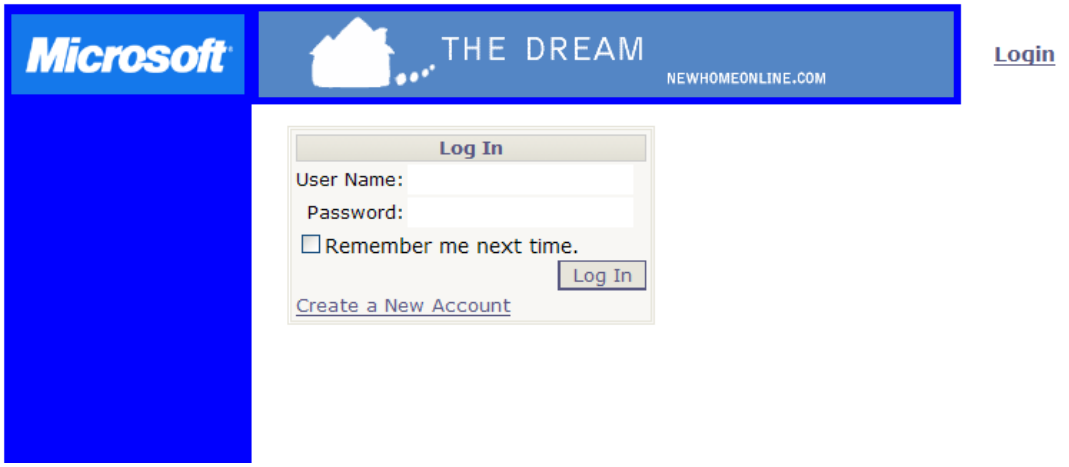
Login.aspx

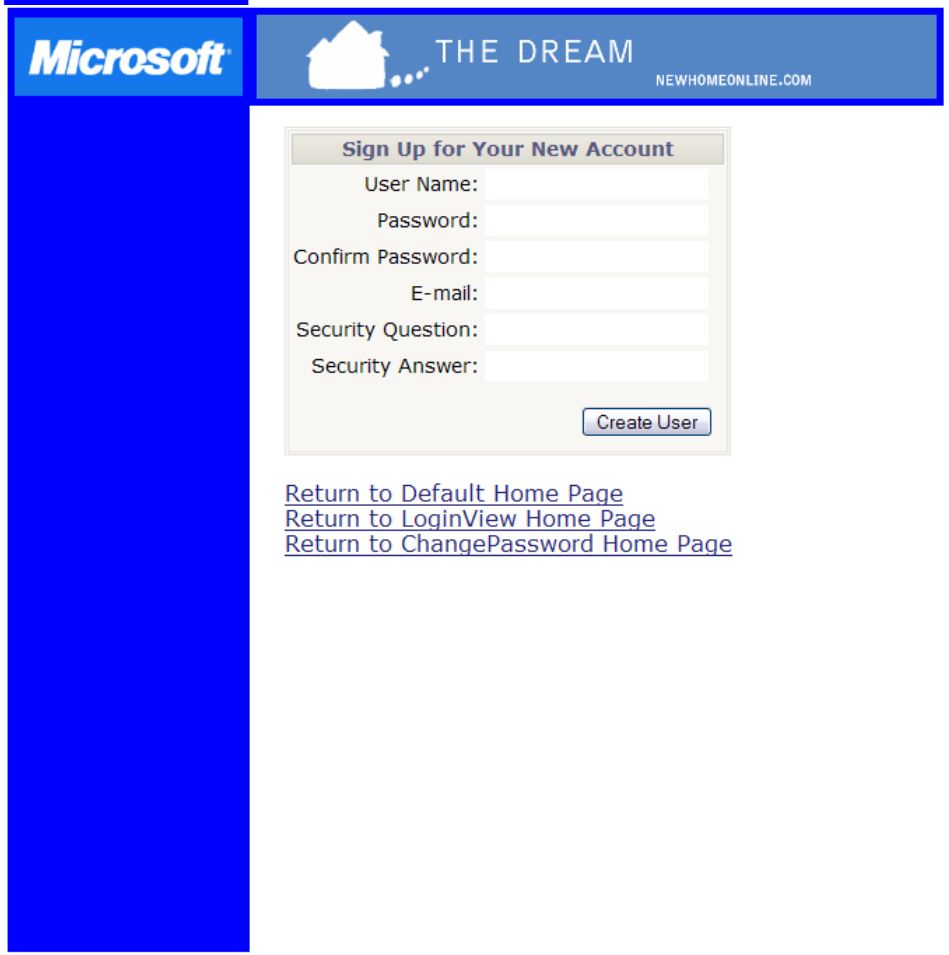
```
<%@ Page Language="C#" MasterPageFile="~/Site.master"%>

<asp:Content ID="Content1" ContentPlaceHolderId="MainBody" runat="server">
  <asp:login ID="Login1" runat="server" createuserurl="CreateUser.aspx"
                createusertext="Create a New Account" />
</asp:Content>
```

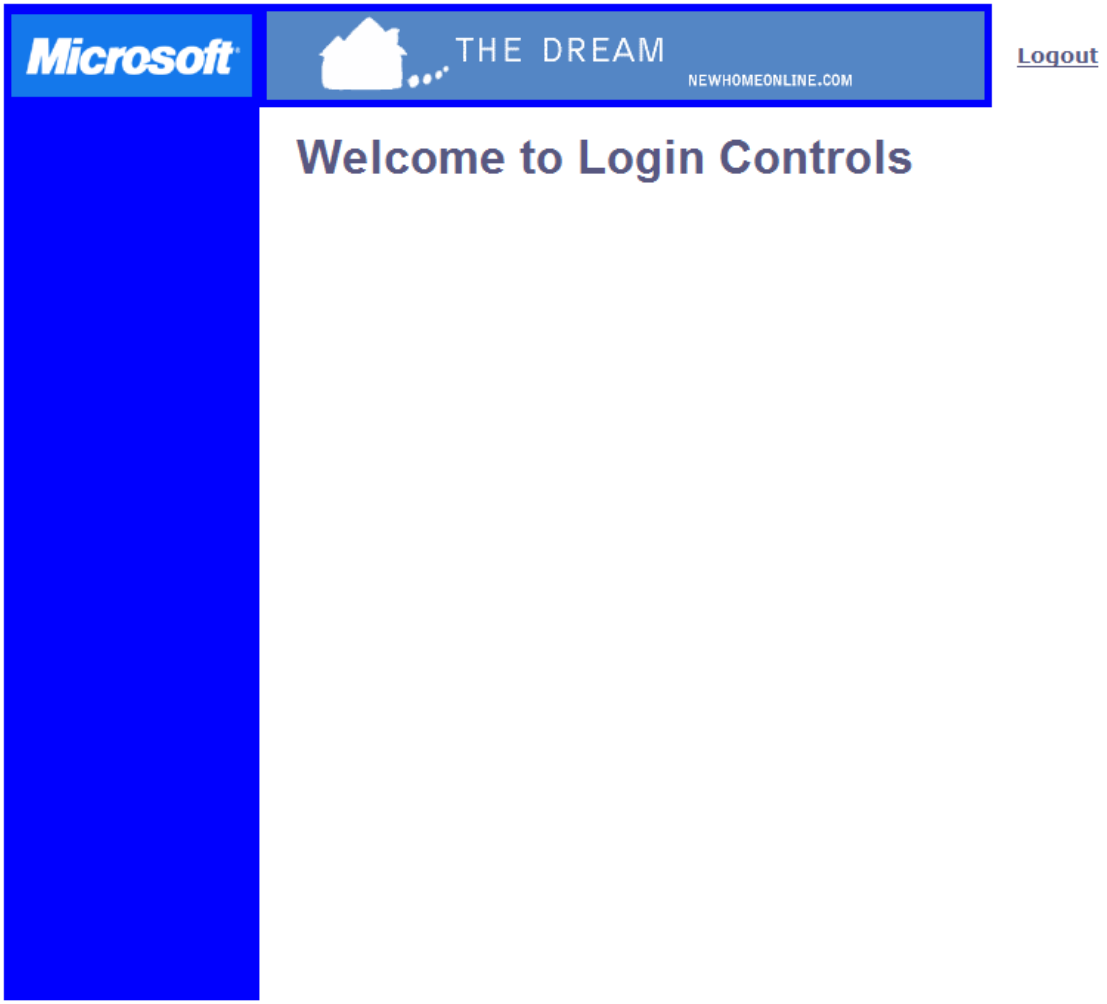
CreateUser.aspx

```
<%@ Page Language="C#" MasterPageFile="~/Site.master"%>

<asp:Content ID="Content1" ContentPlaceHolderId="MainBody" runat="server">
  <asp:CreateUserWizard ID="CreateUserWizard1" runat="server"
          continuedestinationpageurl="Home.aspx"/><br />
  <a href="Home.aspx">Return to Default Home Page</a><br />
  <a href="HomeLoginView.aspx">Return to LoginView Home Page</a><br />
  <a href="HomeChangePassword.aspx">Return to ChangePassword Home Page</a><br />
</asp:Content>
```


Microsoft
THE DREAM
NEWHOMEONLINE.COM
Welcome to Login Controls


Microsoft
THE DREAM
NEWHOMEONLINE.COM
Login
Log In
User Name:
Password:
Remember me next time.
Log In
Create a New Account


Microsoft
THE DREAM
NEWHOMEONLINE.COM
Sign Up for Your New Account
User Name:
Password:
Confirm Password:
E-mail:
Security Question:
Security Answer:
Create User
Return to Default Home Page
Return to LoginView Home Page
Return to ChangePassword Home Page


Microsoft
THE DREAM
NEWHOMEONLINE.COM
Logout
Welcome to Login Controls

Kontrolki Web Parts

- Umożliwiają edycję strony z poziomu przeglądarki
- Każdy użytkownik może dostoswać stronę do swoich upodobań
- Każda kontrolka może być kontrolką Web Part
- Strona budowana jest w oparciu o kontrolki Web Parts, strefy Web Parts Zones oraz kontrolkę WebPartManager, koordynującą współpracę między Web Parts i Web Parts Zones

Klasy WebPart i GenericWebPart

- WebPart - klasa bazowa dla tworzenia nowych kontrolek WebParts
- GenericWebPart - klasa opakowująca. Automatycznie opakowuje dowolną kontrolkę serwerową jeśli ta jest umieszczona w Web Parts Zone. Dzięki temu można używać zwykłych kontrolek tak jakby były to kontrolki WebPart. Nie jest nigdy jawnie specyfikowana w kodzie - opakowanie następuje automatycznie podczas wykonywania.

Klasa WebPartZone

- Definiuje region na stronie w którym zawarte są kontrolki WebPart
- Umożliwia rozmieszczenie kontrolek oraz definiuje wspólny interfejs użytkownika dla tych kontrolek (chrome)

Klasa WebPartManager

- Kontroluje WebParts
- Musi być dokładnie jedna kontrolka tego typu na stronie wykorzystującej WebParts
- Współpracuje tylko z zalogowanymi użytkownikami
- Nie zajmuje się kontrolkami znajdującymi się poza Web Part Zones (nawet jeśli są to kontrolki Web Parts)
- Zajmuje się m. in. dodawaniem i usuwaniem kontrolek, zarządzaniem połączeniami między kontrolkami, przesuwaniem kontrolek oraz przełączaniem między widokami strony (np. użytkowym i konstrukcyjnym)

WebParts.aspx

```
<%@ Page Language="C#" %>
<%@ Register Src="WebPartPageMenu.ascx" TagName="WebPartPageMenu" TagPrefix="uc1"
%>
<html>
  <head runat="server">
    <title>Web Part Page</title>
  </head>
  <body>
    <form id="form1" runat="server">
      <div>
        <asp:WebPartManager ID="WebPartManager1" Runat="server"/>
          <uc1:webpartpagemenu id="WebPartPageMenu1" runat="server"></uc1:webpartpagemenu>
      </div>
      <div>
        <table style="width: 100%">
          <tr>
            <td style="width: 100px; height: 100px" valign="top" align="left">
              <asp:WebPartZone ID="WebPartZone1" Runat="server">
                <ZoneTemplate>
                  <asp:Calendar Runat="server" ID="Calendar1"/>
                </ZoneTemplate>
              </asp:WebPartZone>
            </td>
            <td style="width: 100px; height: 100px" valign="top" align="left">
              <asp:WebPartZone ID="WebPartZone2" Runat="server">
              </asp:WebPartZone>
            </td>
            <td style="width: 100px; height: 100px" valign="top" align="left">
            </td>
          </tr>
        </table>
        <br />
        <asp:LoginStatus ID="LoginStatus1" Runat="server" />
      </div>
    </form>
  </body>
</html>
```

```
<%@ Control Language="C#" ClassName="WebPartPageMenu" %>
<script runat="server">
  WebPartManager _manager;
  void Page_Init(object sender, EventArgs e)
  {
    Page.InitComplete += new EventHandler(InitComplete);
  }
  void InitComplete(object sender, System.EventArgs e)
  {
    _manager = WebPartManager.GetCurrentWebPartManager(Page);
    String browseModeName = WebPartManager.BrowseDisplayMode.Name;
    foreach (WebPartDisplayMode mode in _manager.SupportedDisplayModes)
    {
      String modeName = mode.Name;
      if (mode.IsEnabled(_manager))
      {
        ListItem item = new ListItem(modeName , modeName);
        DisplayModeDropdown.Items.Add(item);
      }
    }
  }

  void DisplayModeDropdown_SelectedIndexChanged(object sender, EventArgs e)
  {
    String selectedMode = DisplayModeDropdown.SelectedValue;
    WebPartDisplayMode mode = _manager.SupportedDisplayModes[selectedMode];
    if (mode != null)
      _manager.DisplayMode = mode;
  }
  void Page_PreRender(object sender, EventArgs e)
  {
    DisplayModeDropdown.SelectedValue = _manager.DisplayMode.Name;
  }

</script>
<div>
  <asp:DropDownList ID="DisplayModeDropdown"
    runat="server"
    AutoPostBack="true"
    EnableViewState="false"
    OnSelectedIndexChanged="DisplayModeDropdown_SelectedIndexChanged" />
</div>
```

Browse

Untitled

<	January 2007					>
Sun	**Mon**	**Tue**	**Wed**	**Thu**	**Fri**	**Sat**
31	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	9	10

Logout

Design

WebPartZone1

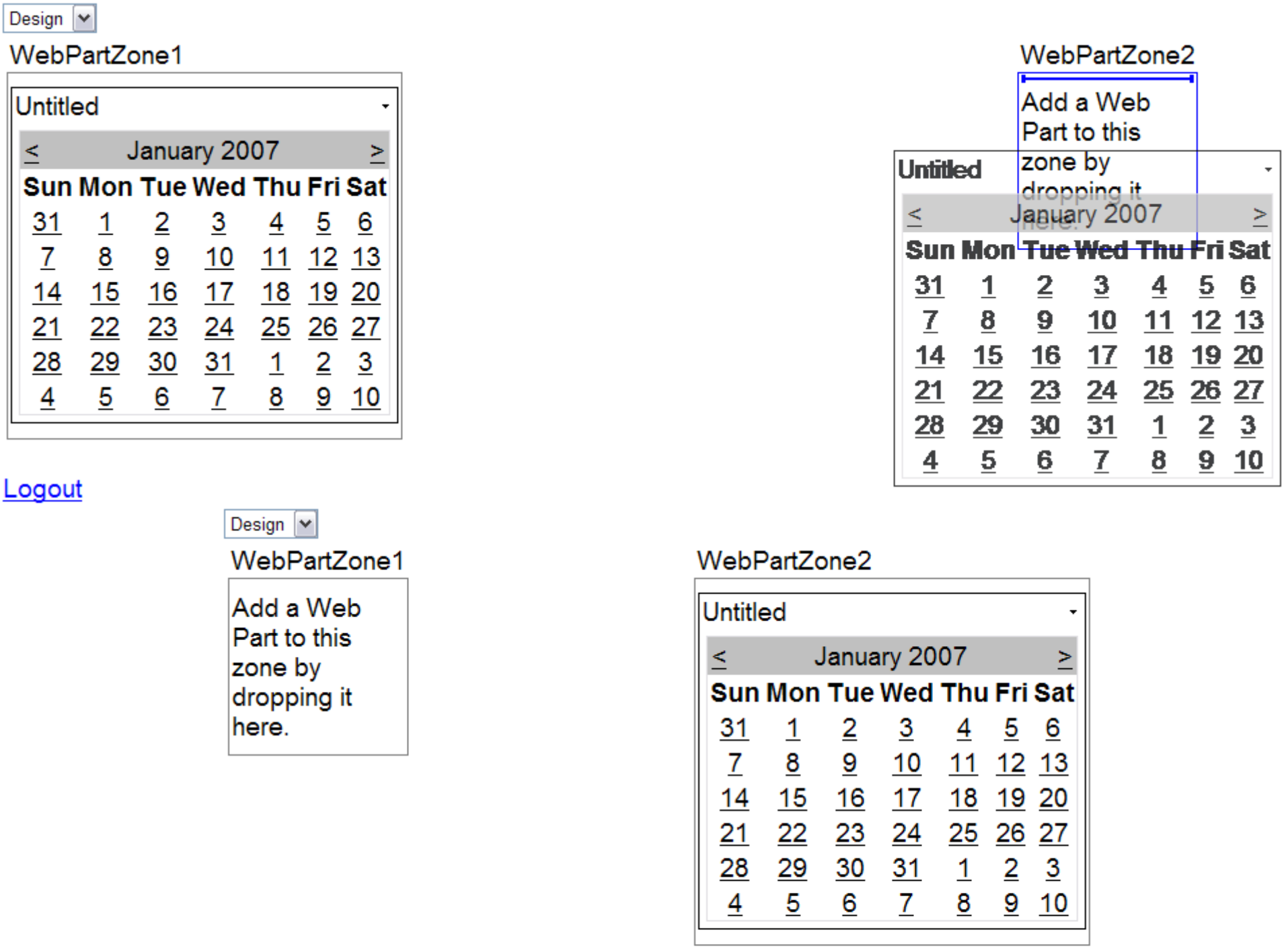
Untitled

<	January 2007					>
Sun	**Mon**	**Tue**	**Wed**	**Thu**	**Fri**	**Sat**
31	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	9	10

WebPartZone2

Add a Web Part to this zone by dropping it here.

Logout


Design
WebPartZone1
Untitled
≤ January 2007 ≥
Sun Mon Tue Wed Thu Fri Sat
31 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 9 10
WebPartZone2
Add a Web Part to this zone by dropping it here.
Untitled
≤ January 2007 ≥
Sun Mon Tue Wed Thu Fri Sat
31 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 9 10
Logout
Design
WebPartZone1
Add a Web Part to this zone by dropping it here.
WebPartZone2
Untitled
≤ January 2007 ≥
Sun Mon Tue Wed Thu Fri Sat
31 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 9 10
Logout

Praca z danymi

- ASP .NET 2.0 zawiera kontrolki umożliwiające łączenie z wieloma źródłami danych:
  - bazą danych SQL (SqlDataSource)
  - obiektem pośredniczącym zapewniającym dostęp do danych (ObjectDataSource)
  - bazą danych MS Access (AccessDataSource)
  - strukturą strony ASP .NET (SiteMapDataSource)
  - plikiem XML (XmlDataSource)

Praca z danymi

- ASP .NET 2.0 zawiera również kontrolki umożliwiające prezentację i obróbkę danych
  - w formie tabeli (GridView)
  - pojedynczy rekord w formie tabeli nazwa-wartość (DetailsView)
  - pojedynczy rekord w sposób zdefiniowany przez wzorzec (FormView)
  - w formie drzewa (TreeView)
  - w formie menu (MenuView)

GridView i SqlDataSource

- Najprostsza forma przedstawienia danych jest tabela umożliwiająca jedynie odczyt, nie pozwalająca na manipulację danymi
- Aby wygenerować tabelę należy najpierw skonfigurować SqlDataSource, a następnie podłączyć GridView ustawiając odpowiednie DataSourceID
- Przy pomocy ConnectionString oraz SelectCommand należy podać komendy umożliwiające połączenie z bazą danych oraz wykonanie zapytania do bazy

```
<%@ Page Language="C#" %>
<html>
<head runat="server">
  <title>GridView Bound to SqlDataSource</title>
</head>
<body>
  <form id="form1" runat="server">
    <asp:GridView ID="GridView1" DataSourceID="SqlDataSource1" runat="server" />
    <asp:SqlDataSource ID="SqlDataSource1" runat="server"
      SelectCommand="SELECT [au_id], [au_lname], [au_fname], [phone], [address], [city],
       [state], [zip], [contract] FROM [authors]"
      ConnectionString="<%$ ConnectionStrings:Pubs %>" />
  </form>
</body>
</html>
```

```
<?xml version="1.0"?>
<configuration>
 <connectionStrings>
    <add name="Pubs" connectionString="Server=(local)\SQLExpress;Integrated
            Security=True;Database=pubs;Persist Security Info=True"
      providerName="System.Data.SqlClient" />
 </connectionStrings>
    <system.web>
        <pages styleSheetTheme="Default"/>
        <caching>
            <sqlCacheDependency enabled="true" pollTime="1000">
                <databases>
                    <add name="Pubs" connectionStringName="Pubs"/>
                </databases>
            </sqlCacheDependency>
        </caching>
        </system.web>
</configuration>
```

GridView i SqlDataSource

- Można określić które z kolumn tabeli będą wyświetlane, w jakiej kolejności i jak zostaną nazwane

```
<%@ Page Language="C#" %>
<html>
<head id="Head1" runat="server">
  <title>GridView Bound Fields</title>
</head>
<body>
  <form id="form1" runat="server">
    <asp:GridView ID="GridView1" DataSourceID="SqlDataSource1" AutoGenerateColumns="False"
      runat="server">
      <Columns>
        <asp:BoundField HeaderText="ID" DataField="au_id" ReadOnly="true" />
        <asp:BoundField HeaderText="Last Name" DataField="au_lname" />
        <asp:BoundField HeaderText="First Name" DataField="au_fname" />
        <asp:BoundField HeaderText="Phone" DataField="phone" />
        <asp:BoundField HeaderText="Address" DataField="address" />
        <asp:CheckBoxField HeaderText="Contract" DataField="contract" />
      </Columns>
    </asp:GridView>
    <asp:SqlDataSource ID="SqlDataSource1" runat="server"
      SelectCommand="SELECT [au_id], [au_lname], [au_fname], [phone], [address], [city], [state],
                     [zip], [contract] FROM [authors]"
      ConnectionString="<%$ ConnectionStrings:Pubs %>" />
  </form>
</body>
</html>
```

GridView i SqlDataSource

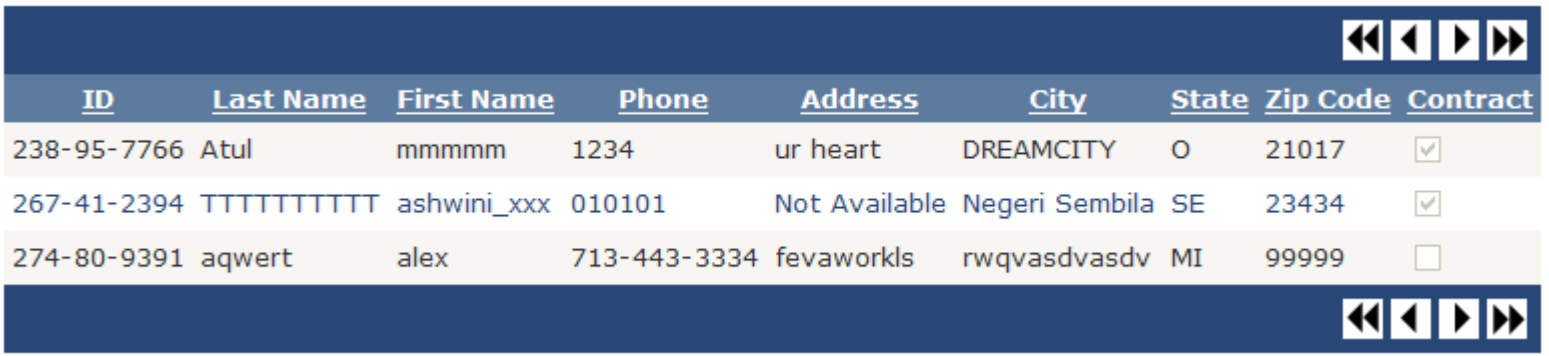
- Można umożliwić sortowanie danych w kolumnach. Nazwy zostaną wtedy wyświetlone jako linki
- Kolejne kliknięcia w dany link przełączają między sortowaniem rosnąco i malejąco

```
<asp:GridView ID="GridView1" AllowSorting="true" runat="server"
    DataSourceID="SqlDataSource1"   AutoGenerateColumns="False">
```

GridView i SqlDataSource

- Można podzielić tabelę na strony. Umożliwia to własność AllowPaging
- Domyślnie wyświetlane jest 10 rekordów na stronie, ale można to zmienić polem PageSize
- Poprzez własność PagerSettings można określić wygląd paska umożliwiającego przełączanie między stronami
- Poprzez własność PagerStyle można określić styl paska przełączającego

```
<asp:GridView ID="GridView1" AllowSorting="true" AllowPaging="true" PageSize="3"
        runat="server" DataSourceID="SqlDataSource1" AutoGenerateColumns="False">
  <PagerSettings Mode="NextPreviousFirstLast" Position="TopAndBottom"
        FirstPageImageUrl="~/Images/First.gif" LastPageImageUrl="~/Images/Last.gif"
        NextPageImageUrl="~/Images/Next.gif" PreviousPageImageUrl="~/Images/Prev.gif" />
  <PagerStyle ForeColor="White" HorizontalAlign="Right" BackColor="#284775" />
```



ID	Last Name	First Name	Phone	Address	City	State	Zip Code	Contract
238-95-7766	Atul	mmmmm	1234	ur heart	DREAMCITY	O	21017	☑
267-41-2394	TTTTTTTTTT	ashwini_xxx	010101	Not Available	Negeri Sembila	SE	23434	☑
274-80-9391	aqwert	alex	713-443-3334	fevaworkls	rwqvasdvasdv	MI	99999	☐

GridView i SqlDataSource

- O ile źródło danych to umożliwia, GridView pozwala na edycję danych (poprawianie i usuwanie)
- Można to osiągnąć
  - ustawiając własności AutoGenerateEditButton oraz AutoGenerateDeleteButton, lub
  - dodając CommandField i ustawiając w nim ShowEditButton i ShowDeleteButton

```
<%@ Page Language="C#" %>
<html>
  <head runat="server">
    <title>Updating Data Using GridView</title>
  </head>
  <body>
    <form id="form1" runat="server">
      <asp:GridView ID="GridView1" AllowSorting="true" AllowPaging="true" Runat="server"
        DataSourceID="SqlDataSource1" AutoGenerateEditButton="true" DataKeyNames="au_id"
        AutoGenerateColumns="False">
```

	ID	Last Name	First Name	Phone	Address	City	State	Zip Code	Contract
Edit	111-12-5265	9879879	harry	546-8845	228 s mass	lake	kk	11111	☑
Edit	172-32-1176	test	yyyy	12345678	West Park Street	Dewsbury	UK	11111	☐
Edit	213-46-8915	Hell bery	jim	883919128	wellll new year !	Somewhere	VA	89880	☑
Edit	238-95-7766	Atul	mmmmm	1234	ur heart	DREAMCITY	O	21017	☑
Edit	267-41-2394	TTTTTTTTTT	ashwini_xxx	010101	Not Available	Negeri Sembila	SE	23434	☑
Edit	274-80-9391	aqwert	alex	713-443-3334	fevaworkls	rwqvasdvasdv	MI	99999	☐
Edit	409-56-7008	Patel	RAMASCHANDRAN	713-867-666	ON THE RULES OF THE ROAD	Nijmegen	NG	77063	☑
Edit	427-17-2319	John	BOND1	888888	Happy	Campinas - Brazil1`2	BI	13100	☑
Edit	472-27-2349	Morschhauuser2	25522	wqe	kira	54s6d4asd	VA	11111	☐
Edit	486-29-1786	oombu	olug	kandaraoli	punda	sunni	TN	76092	☑

1 2 3

	ID	Last Name	First Name	Phone	Address	City	State	Zip Code	Contract
Update Cancel	111-12-5265	9879879	harry	546-8845	228 s mass	lake	kk	11111	☑
Edit	172-32-1176	test	yyyy	12345678	West Park Street	Dewsbury	UK	11111	☐
Edit	213-46-8915	Hell bery	jim	883919128	wellll new year !	Somewhere	VA	89880	☑

GridView i SqlDataSource

- GridView umożliwia filtrowanie danych
- Można się posłużyć specjalnymi parametrami, np. QueryStringParameter lub ControlParameter

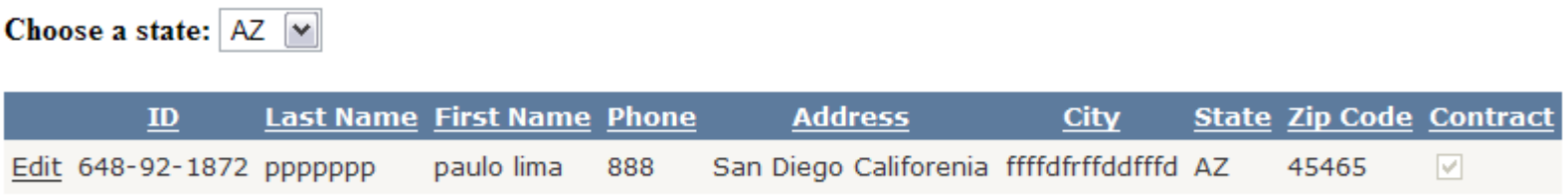
```
<asp:SqlDataSource ID="SqlDataSource1" Runat="server" SelectCommand="SELECT [au_id], [au_lname], [au_fname], [phone], [address], [city], [state], [zip], [contract] FROM [authors] WHERE [state] = @state"
      <SelectParameters>
        <asp:QueryStringParameter Name="state" QueryStringField="state"
          DefaultValue="CA" />
      </SelectParameters>
    </asp:SqlDataSource>
```

http://.../GridViewQueryString_cs.aspx?state=IN

	ID	Last Name	First Name	Phone	Address	City	State	Zip Code	Contract
Edit	274-80-9391	1111	kawaii	713-443-3334		Koit x	IN	99999	☐

GridView i SqlDataSource

```
<form id="form1" runat="server">
    <b>Choose a state:</b>
    <asp:DropDownList ID="DropDownList1" DataSourceID="SqlDataSource2"
          AutoPostBack="true"
      DataTextField="state" Runat="server" />
    <asp:SqlDataSource ID="SqlDataSource2" Runat="server" SelectCommand="SELECT
     DISTINCT [state] FROM [authors]"
      ConnectionString="<%$ ConnectionStrings:Pubs %>" />


<asp:SqlDataSource ID="SqlDataSource1" Runat="server" SelectCommand="SELECT [au_id],
[au_lname], [au_fname], [phone], [address], [city], [state], [zip], [contract] FROM [authors] WHERE
[state] = @state"
      ConnectionString="<%$ ConnectionStrings:Pubs %>">
      <SelectParameters>
        <asp:ControlParameter Name="state" ControlID="DropDownList1" />
      </SelectParameters>
    </asp:SqlDataSource>
</form>
```



GridView i SqlDataSource

- Możliwe jest ustawienie cache dla SqlDataSource
- Dzięki temu przy kolejnych odwołaniach do określonych danych nie będą one pobierane z bazy danych, tylko z lokalnej kopii
- Można okreslić czas życia danych w cache, można też określić CacheExpirationPolicy - czy odczyt danych powoduje reset licznika czasu życia (sliding) czy nie (absolute)

```
<asp:SqlDataSource ID="SqlDataSource1" Runat="server" SelectCommand="SELECT
DatePart(second, GetDate()) As TimeStamp, [au_id], [au_lname], [au_fname], [phone], [address],
[city], [state], [zip], [contract] FROM [authors] WHERE [state] = @state"
        UpdateCommand="UPDATE [authors] SET [au_lname] = @au_lname, [au_fname] =
@au_fname, [phone] = @phone, [address] = @address, [city] = @city, [state] = @state, [zip] =
@zip, [contract] = @contract WHERE [au_id] = @au_id"
        ConnectionString="<%$ ConnectionStrings:Pubs %>" CacheDuration="5"
      EnableCaching="True">
```

GridView i SqlDataSource

- Innym rozwiązaniem jest załadowanie wszystkich danych do cache i przeprowadzanie filtrowania na cache
- W tym celu można użyć własności FilterExpression i kolekcji FilterParameters

```
    <asp:DropDownList ID="DropDownList1" DataSourceID="SqlDataSource2"
AutoPostBack="true"
      DataTextField="state" Runat="server" />
    <asp:SqlDataSource ID="SqlDataSource2" Runat="server" SelectCommand="SELECT
DISTINCT [state] FROM [authors]"
      ConnectionString="<%$ ConnectionStrings:Pubs %>" />

    <asp:SqlDataSource ID="SqlDataSource1" Runat="server" SelectCommand="SELECT
DatePart(second, GetDate()) As TimeStamp, [au_id], [au_lname], [au_fname], [phone], [address],
[city], [state], [zip], [contract] FROM [authors]"
      ConnectionString="<%$ ConnectionStrings:Pubs %>" CacheDuration="5"
EnableCaching="True" FilterExpression="state='{0}'">
      <FilterParameters>
        <asp:ControlParameter ControlID="DropDownList1" Name="state"
          PropertyName="SelectedValue" />
      </FilterParameters>
```

GridView i SqlDataSource

- Jeśli korzysta się z Microsoft SQL Server można ustawić tryb pracy cache w którym dane w cache są przechowywane tak długo, aż nie nastąpi zmiana danych na serwerze
- Umożliwia to wydłużenie czasu życia danych w cache bez obawy prezentacji uzytkownikowi danych nieaktualnych
- Dla SQL Server 2005 następuje powiadomienie o zmianie danych, dla wcześniejszych wersji przeprowadzane jest odpytywanie
- Używa się własności SqlCacheDependency. Dla powiadamiania ustawia się ją na "CommandNotification", dla pollingu na "nazwa_połączenia_z_DB:nazwa_tabeli"

GridView i SqlDataSource

```
<asp:SqlDataSource ID="SqlDataSource1" Runat="server" SelectCommand="SELECT
DatePart(second, GetDate()) As TimeStamp, [au_id], [au_lname], [au_fname], [phone], [address],
[city], [state], [zip], [contract] FROM [authors]"
        ConnectionString="<%$ ConnectionStrings:Pubs %>" EnableCaching="True"
            FilterExpression="state='{0}'" SqlCacheDependency="Pubs:Authors">
        <FilterParameters>
          <asp:ControlParameter ControlID="DropDownList1" Name="state"
            PropertyName="SelectedValue" />
        </FilterParameters>
      </asp:SqlDataSource>
```

GridView, DetailsView i SqlDataSource

- Można tak skonfigurować dostępdo danych, aby kontrolka DeteilsView wyświetlała szczegóły rekordu wybranego w GridView
- W tem celu można użyć własności SelectedValue która ma warość pierwszego pola wybranego rekordu
- Następnie ta własność może być użyta jako ControlParameter w zapytaniu do bazy, którego wynik przedstawi DetailsView

```
<body>
  <form id="form1" runat="server">
    <b>Choose a state:</b>
    <asp:DropDownList ID="DropDownList1" DataSourceID="SqlDataSource2" AutoPostBack="true"
      DataTextField="state" runat="server" />
    <asp:SqlDataSource ID="SqlDataSource2" runat="server" SelectCommand="SELECT
                  DISTINCT [state] FROM [authors]"
      ConnectionString="<%$ ConnectionStrings:Pubs %>" />
    <table>
      <tr>
        <td valign="top">
          <asp:GridView ID="GridView1" AllowSorting="True" AllowPaging="True" runat="server"
            DataSourceID="SqlDataSource1" DataKeyNames="au_id"
            AutoGenerateColumns="False" Width="427px">
            <Columns>
              <asp:CommandField ShowSelectButton="True" />
              <asp:BoundField DataField="au_id" HeaderText="au_id" ReadOnly="True"
                            SortExpression="au_id" />
              <asp:BoundField DataField="au_lname" HeaderText="au_lname"
                            SortExpression="au_lname" />
              <asp:BoundField DataField="au_fname" HeaderText="au_fname"
                            SortExpression="au_fname" />
              <asp:BoundField DataField="state" HeaderText="state" SortExpression="state" />
            </Columns>
          </asp:GridView>
          <asp:SqlDataSource ID="SqlDataSource1" runat="server" SelectCommand="SELECT
                        [au_id], [au_lname], [au_fname], [state] FROM [authors] WHERE ([state] =
@state)"
            ConnectionString="<%$ ConnectionStrings:Pubs %>">
            <SelectParameters>
              <asp:ControlParameter ControlID="DropDownList1" Name="state"
                            PropertyName="SelectedValue" Type="String" />
            </SelectParameters>
          </asp:SqlDataSource>
        </td>
```

```
      <td valign="top">
        <asp:DetailsView AutoGenerateRows="False" DataKeyNames="au_id"
                         DataSourceID="SqlDataSource3"
          HeaderText="Author Details" ID="DetailsView1" runat="server" Width="275px">
          <Fields>
            <asp:BoundField DataField="au_id" HeaderText="au_id" ReadOnly="True"
                            SortExpression="au_id" />
            <asp:BoundField DataField="au_lname" HeaderText="au_lname"
                            SortExpression="au_lname" />
            <asp:BoundField DataField="au_fname" HeaderText="au_fname"
                            SortExpression="au_fname" />
            <asp:BoundField DataField="phone" HeaderText="phone" SortExpression="phone" />
            <asp:BoundField DataField="address" HeaderText="address" SortExpression="address"
                 />
            <asp:BoundField DataField="city" HeaderText="city" SortExpression="city" />
            <asp:BoundField DataField="state" HeaderText="state" SortExpression="state" />
            <asp:BoundField DataField="zip" HeaderText="zip" SortExpression="zip" />
            <asp:CheckBoxField DataField="contract" HeaderText="contract"
                            SortExpression="contract" />
          </Fields>
        </asp:DetailsView>
        <asp:SqlDataSource ConnectionString="<%$ ConnectionStrings:Pubs %>"
                           ID="SqlDataSource3"
          runat="server" SelectCommand="SELECT [au_id], [au_lname], [au_fname], [phone],
                [address], [city], [state], [zip], [contract] FROM [authors] WHERE ([au_id] =
@au_id)">
          <SelectParameters>
            <asp:ControlParameter ControlID="GridView1" Name="au_id"
                            PropertyName="SelectedValue" Type="String" />
          </SelectParameters>
        </asp:SqlDataSource>
      </td>
    </tr>
  </table>
  <br />
 </form>
</body>
```

Choose a state: CO

	au_id	au_lname	au_fname	state
Select	427-17-2319	Is Great99999999	CptKirk	CO

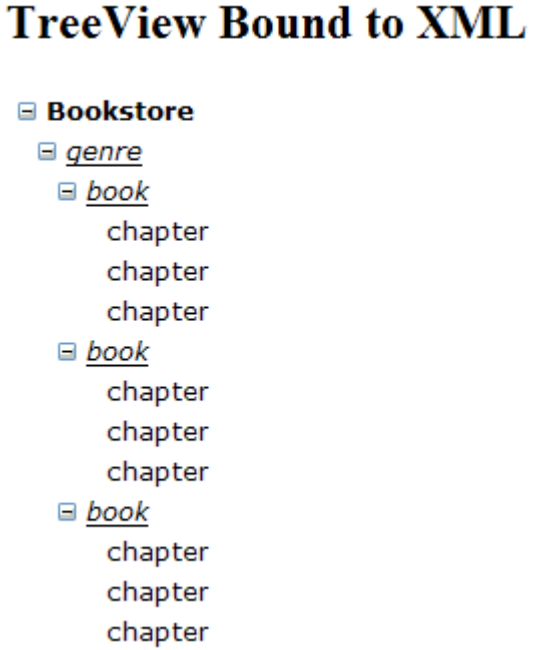
Author Details	
au_id	427-17-2319
au_lname	Is Great99999999
au_fname	CptKirk
phone	212-555-5555
address	12366336
city	Campinas - Brazil12
state	CO
zip	54321
contract	☑

TreeView i XmlDataSource

- Możliwe jest przedstawianie danych hierarchicznych (np. dokumentu XML) w postaci drzewa
- Domyślnie wyświetlana jest nazwa węzła dokumentu XML, co nie zawsze ma sens

```
<Bookstore>
  <genre name="Business">
    <book ISBN="BU1032" Title="The Busy Executive's Database Guide" Price="19.99">
      <chapter num="1" name="Introduction">
        Abstract...
      </chapter>
      <chapter num="2" name="Body">
        Abstract...
      </chapter>
      <chapter num="3" name="Conclusion">
        Abstract...
      </chapter>
    </book>
    <book ISBN="BU2075" Title="You Can Combat Computer Stress!" Price="2.99">
      <chapter num="1" name="Introduction">
        Abstract...
      </chapter>
      <chapter num="2" name="Body">
        Abstract...
      </chapter>
      <chapter num="3" name="Conclusion">
        Abstract...
      </chapter>
    </book>
    <book ISBN="BU7832" Title="Straight Talk About Computers" Price="19.99">
      <chapter num="1" name="Introduction">
        Abstract...
      </chapter>
      <chapter num="2" name="Body">
        Abstract...
      </chapter>
      <chapter num="3" name="Conclusion">
        Abstract...
      </chapter>
    </book>
  </genre>
</Bookstore>
```

```
<%@ Page Language="C#" Theme="Default" %>

<html xmlns="http://www.w3.org/1999/xhtml">
  <head runat="server">
    <title>TreeView Bound to XML</title>
  </head>
  <body>
    <div>
      <h2>TreeView Bound to XML</h2>

      <form action="treeview12_xml_cs.aspx" runat="server">

        <asp:XmlDataSource ID="MySource" DataFile="~/App_Data/Bookstore.xml"
            runat="server"/>

        <asp:TreeView ID="TreeView1"
          SkinId="Bookstore"
          DataSourceId="MySource"
          ExpandDepth="3"
          MaxDataBindDepth="3"
          runat="server" />

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

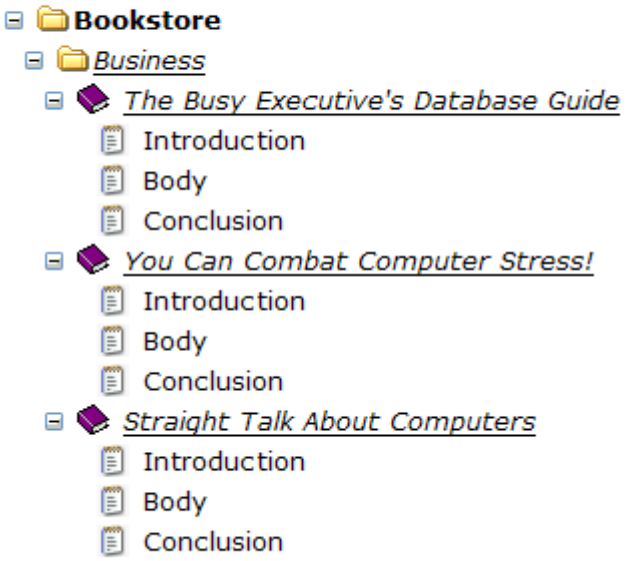


TreeView i XmlDataSource

- Aby uzyskać lepsze przedstawieniei dokumentu XML, można użyć obiektów TreeNodeBinding
- Mają one własność DataMember, wskazującą nawę elementu którego dotyczą, oraz Depth - wskazującą poziom hierarchii którego dotyczą. Można ustawić oba
- W TreeNodeBinding można wskazać co będzie wyświetlane. Np. własność TextField określa nazwę atrybutu która będzie wypisana jako nawa węzła drzewa, ImageUrl - nazwę pliku graficznego wyświetlanego w węźle
- Elementy TreeNodeBinding są umieszczane w kolekcji Databindings

```
<%@ Page Language="C#" Theme="Default" %>

<html xmlns="http://www.w3.org/1999/xhtml">
  <head runat="server">
    <title>TreeView Bound to XML</title>
  </head>
  <body>
    <div>
      <h2>TreeView Bound to XML</h2>

      <form action="treeview13_xml_cs.aspx" runat="server">

        <asp:XmlDataSource ID="MySource" DataFile="~/App_Data/Bookstore.xml" runat="server"/>

        <asp:TreeView ID="TreeView1"
          SkinId="Bookstore"
          DataSourceId="MySource"
          ExpandDepth="3"
          MaxDataBindDepth="3"
          runat="server">
          <Databindings>
            <asp:TreeNodeBinding DataMember="Bookstore" Text="Bookstore"
              ImageUrl="~/images/xp/folder.gif" />
            <asp:TreeNodeBinding DataMember="genre" TextField="name"
              ImageUrl="~/images/xp/folder.gif" />
            <asp:TreeNodeBinding DataMember="book" TextField="Title"
              ImageUrl="~/images/chm/closedbook.gif" />
            <asp:TreeNodeBinding DataMember="chapter" TextField="name"
              ImageUrl="~/images/xp/notepad.gif" />
          </Databindings>

        </asp:TreeView>

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

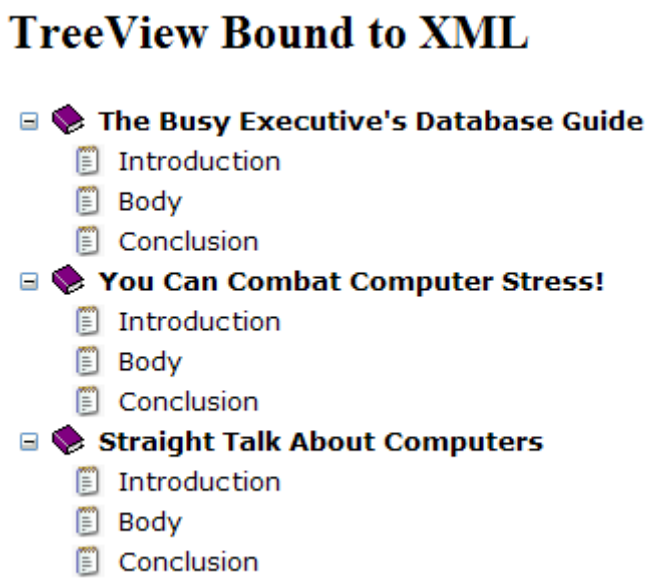
TreeView Bound to XML



TreeView i XmlDataSource

- XmlDataSource ma własność XPath umożliwiającą filtrowanie danych wyświetlanych przez TreeView

```
<asp:XmlDataSource ID="MySource" DataFile="~/App_Data/Bookstore.xml"
    XPath="Bookstore/genre[@name='Business']/book" runat="server"/>
```



DataList

- Wyświetla dabe jako listę, opcjonalnie umożliwiając edycję elementów
- Zawartość i sposób wyświetlania są określone poprzez wzorce (templates), lista musi co najmniej definiować ItemTemplate
- Inne to AlternatingItemTemplate - określa 'co drugi' element, SeparatorTemplate - element rozdzielający, SelectedItemTemplate - element wybrany, HeaderTemplate, FooterTemplate - nagłówek i stopka
- RepeatLayout ustawiony na Flow usuwa stukturę tabeli z listy
- RepeatDirection określa kierunek (poziomy/pionowy) a RepeatColumns liczbę kolumn

```
<%@ Import Namespace="System.Data" %>
<html>
<script language = "C#" runat="server">
    ICollection CreateDataSource() {
        DataTable dt = new DataTable();
        DataRow dr;
        dt.Columns.Add(new DataColumn("StringValue", typeof(string)));
        for (int i = 0; i < 10; i++) {
            dr = dt.NewRow();
            dr[0] = "Item " + i.ToString();
            dt.Rows.Add(dr);
        }
        DataView dv = new DataView(dt);
        return dv;
    }

    void Page_Load(Object Sender, EventArgs E) {
        if (!IsPostBack) {
            DataList1.DataSource = CreateDataSource();
            DataList1.DataBind();
        }
    }
    void Button1_Click(Object Sender, EventArgs E) {
        if (DropDown1.SelectedIndex == 0)
            DataList1.RepeatDirection = RepeatDirection.Horizontal;
        else
            DataList1.RepeatDirection = RepeatDirection.Vertical;
        if (DropDown2.SelectedIndex == 0)
            DataList1.RepeatLayout = RepeatLayout.Table;
        else
            DataList1.RepeatLayout = RepeatLayout.Flow;
        DataList1.RepeatColumns=DropDown3.SelectedIndex+1;

        if ((Check1.Checked ==true) && (DataList1.RepeatLayout == RepeatLayout.Table)) {
            DataList1.BorderWidth = Unit.Pixel(1);
            DataList1.GridLines = GridLines.Both;
        }
        else  {
            DataList1.BorderWidth = Unit.Pixel(0);
            DataList1.GridLines = GridLines.None;
        }
    }
</script>
    </form>
</body>
</html>
```

```
<body>
   <h3><font face="Verdana">Simple DataList Sample</font></h3>
   <form runat=server>
   <font face="Verdana" size="-1">
      <asp:DataList id="DataList1" runat="server"
         BorderColor="black"
         CellPadding="3"
         Font-Names="Verdana"
         Font-Size="8pt"
         HeaderStyle-BackColor="#aaaadd"
         AlternatingItemStyle-BackColor="Gainsboro" >
           <HeaderTemplate>
             Items
           </HeaderTemplate>
           <ItemTemplate>
             <%# DataBinder.Eval(Container.DataItem, "StringValue") %>
           </ItemTemplate>
      </asp:DataList>
      <br /><br />
      <hr noshade align="left" width="300px">
      RepeatDirection:
      <asp:DropDownList id=DropDown1 runat="server">
         <asp:ListItem>Horizontal</asp:ListItem>
         <asp:ListItem>Vertical</asp:ListItem>
      </asp:DropDownList><br>

      RepeatLayout:
      <asp:DropDownList id=DropDown2 runat="server">
         <asp:ListItem>Table</asp:ListItem>
         <asp:ListItem>Flow</asp:ListItem>
      </asp:DropDownList><br>

      RepeatColumns:
      <asp:DropDownList id=DropDown3 runat="server">
         <asp:ListItem>1</asp:ListItem>
         <asp:ListItem>2</asp:ListItem>
         <asp:ListItem>3</asp:ListItem>
         <asp:ListItem>4</asp:ListItem>
         <asp:ListItem>5</asp:ListItem>
      </asp:DropDownList><br>

      Show Borders:
      <asp:CheckBox id=Check1 runat="server" /><p>

      <asp:LinkButton id=Button1 Text="Refresh DataList" OnClick="Button1_Click" runat="server"/>
   </font>
</form>
</body>
</html>
```

Simple DataList Sample

Items
Item 0
Item 1
Item 2
Item 3
Item 4
Item 5
Item 6
Item 7
Item 8
Item 9

RepeatDirection: Horizontal

RepeatLayout: Table

RepeatColumns: 1

Show Borders:

Refresh DataList

AdRotator

- Umożliwia wyświetlanie losowych obrazów
- Kliknięcie przenosi do określonej strony
- Opis reklam zawarty jest w pliku XML
- Każda reklama musi zawierać ImageUrl - adres obrazu oraz może zawierać:
  - NavigateUrl - adres po kliknięciu
  - AlternateText - atrybut ALT
  - Keyword - kategoria reklamy
  - Impressions - 'waga' reklamy, determinuje jak często w stosunku do innych będzie dana reklama wyświetlana (im większa wartość, tym częściej)

```
<html>
<body>
    <h3><font face="Verdana">AdRotator Example</font></h3>
    <form runat=server>
        <asp:AdRotator id="ar1" AdvertisementFile="Ads.xml" BorderWidth="1"
                runat=server />
    </form>
</body>
</html>
```

Ads.xml

```
<Advertisements>

  <Ad>
    <ImageUrl>images/banner1.gif</ImageUrl>
    <NavigateUrl>http://www.microsoft.com</NavigateUrl>
    <AlternateText>Alt Text</AlternateText>
    <Keyword>Computers</Keyword>
    <Impressions>80</Impressions>
  </Ad>

  <Ad>
    <ImageUrl>images/banner2.gif</ImageUrl>
    <NavigateUrl>http://www.microsoft.com</NavigateUrl>
    <AlternateText>Alt Text</AlternateText>
    <Keyword>Computers</Keyword>
    <Impressions>80</Impressions>
  </Ad>

  <Ad>
    <ImageUrl>images/banner3.gif</ImageUrl>
    <NavigateUrl>http://www.microsoft.com</NavigateUrl>
    <AlternateText>Alt Text</AlternateText>
    <Keyword>Computers</Keyword>
    <Impressions>80</Impressions>
  </Ad>

</Advertisements>
```

FileUpload

- Umożliwia wybór pliku na komputerze lokalnym i wysłanie go na serwer
- Kontrolka nie zawiera przycisku 'wyślij', należy go dodać samodzielnie

```
<html>
<head>

    <script language="C#" runat="server">

    void Button1_Click(object sender, EventArgs e)
    {
        if (FileUpLoad1.HasFile)
        {
            FileUpLoad1.SaveAs("C:\SomePhysicalPath" + FileUpLoad1.Filename);
            Label1.Text = "Received " + FileUpLoad1.FileName + " Content Type " +
              FileUpLoad1.PostedFile.ContentType + " Length " +
              FileUpLoad1.PostedFile.ContentLength;
        }
        else
        {
            Label1.Text = "No uploaded file";
        }

    }

    </script>

</head>
<body>

    <h3><font face="Verdana">File Upload</font></h3>

    <form runat=server>

        <asp:FileUpLoad id="FileUpLoad1" AlternateText="You cannot upload files" runat="server"
                />
        <asp:Button id="Button1" Text="Upload" OnClick="Button1_Click" runat="server" />
        <asp:Label id="Label1" runat="server" />
    </form>

</body>
</html>
```

ImageMap

- Umożliwia określenie obszarów obrazu reagujących na kliknięcia
- W wyniku kliknięcia może nastąpić przejście do innej strony lub uruchomienie fragmentu kodu
- Obszary definiuje się przy pomocy asp:RectangleHotSpot

```
<%@ Page Language="C#" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<script runat="server">
</script>
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <h3><font face="Verdana">ImageMap Class Mixed HotSpotMode Example</font></h3>
            <asp:imagemap id="Buttons" imageurl="hotspot.jpg" alternatetext="Navigate buttons"
                runat="Server">

                <asp:RectangleHotSpot
                hotspotmode="Navigate"
                NavigateUrl="navigate1.htm"
                alternatetext="Button 1"
                top="30"
                left="175"
                bottom="110"
                right="355">
                </asp:RectangleHotSpot>

                <asp:RectangleHotSpot
                hotspotmode="Navigate"
                NavigateUrl="navigate2.htm"
                alternatetext="Button 2"
                top="155"
                left="175"
                bottom="240"
                right="355">
                </asp:RectangleHotSpot>

                <asp:RectangleHotSpot
                hotspotmode="Navigate"
                NavigateUrl="navigate3.htm"
                alternatetext="Button 3"
                top="285"
                left="175"
                bottom="365"
                right="355">
                </asp:RectangleHotSpot>

            </asp:imagemap>
        </div>
    </form>
</body>
</html>
```

```
<%@ Page Language="C#" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<script runat="server">
    void Buttons_Clicked(object sender, ImageMapEventArgs e)
    {
        label1.Text = e.PostBackValue + " clicked!";

    }
</script>
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <h3><font face="Verdana">ImageMap Class Mixed HotSpotMode Example</font></h3>
            <asp:imagemap id="Buttons" imageurl="hotspot.jpg" alternatetext="Navigate buttons"
                hotspotmode="Postback" onclick="Buttons_Clicked" runat="Server">
                <asp:RectangleHotSpot
                hotspotmode="Postback"
                postbackvalue="Button1"
                alternatetext="Button 1"
                top="30"
                left="175"
                bottom="110"
                right="355">
                </asp:RectangleHotSpot>

/*

*/
                <asp:RectangleHotSpot
                hotspotmode="Postback"
                postbackvalue="Background"
                alternatetext="Background"
                top="0"
                left="0"
                bottom="390"
                right="540">
                </asp:RectangleHotSpot>
            </asp:imagemap>
            <p>
                <h3><font face="verdana"><asp:label id="label1" runat="server" ></asp:label></font></h3>
            </p>
        </div>
    </form>
</body>
</html>
```

MultiView

- Jest zasobnikiem dla grupy kontrolek
- Umożliwia wyświetlanie określonej grupy w zależności od zewnętrznych czynników (określony użytkownik, preferencje użytkownika itp.)

```
<%@ Page Language="C#" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<script runat="server">

    protected void DropDownList1_SelectedIndexChanged(object sender, EventArgs e)
    {
        MultiView1.ActiveViewIndex = Convert.ToInt32(DropDownList1.SelectedValue);
    }

</script>
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <h3>
                <font face="Verdana">MultiView with 3 Views</font>
            </h3>
            <asp:DropDownList ID="DropDownList1" runat="server" AutoPostBack="True"
                 OnSelectedIndexChanged="DropDownList1_SelectedIndexChanged">
                <asp:ListItem Value="0">View 1</asp:ListItem>
                <asp:ListItem Value="1">View 2</asp:ListItem>
                <asp:ListItem Value="2">View 3</asp:ListItem>
            </asp:DropDownList><br />
            <hr />
            <asp:MultiView ID="MultiView1" runat="server" ActiveViewIndex="0">
                <asp:View ID="View1" runat="server">
                    Now showing View #1<br />
                    <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox><strong> </strong>
                    <asp:Button ID="Button1" runat="server" Text="Button" /></asp:View>
                <asp:View ID="View2" runat="server">
                    Now showing View #2<br />
                    <asp:HyperLink ID="HyperLink1" runat="server"
                          NavigateUrl="http://www.asp.net">HyperLink</asp:HyperLink>
                    <asp:HyperLink ID="HyperLink2" runat="server"
                          NavigateUrl="http://www.asp.net">HyperLink</asp:HyperLink></asp:View>
                <asp:View ID="View3" runat="server">
                    Now showing View #3<br />
                    <asp:Calendar ID="Calendar1" runat="server"></asp:Calendar>
                </asp:View>
            </asp:MultiView></div>
    </form>
</body>
</html>
```

MultiView with 3 Views

View 1

Now showing View #1

Button

MultiView with 3 Views

View 2

Now showing View #2

HyperLink HyperLink

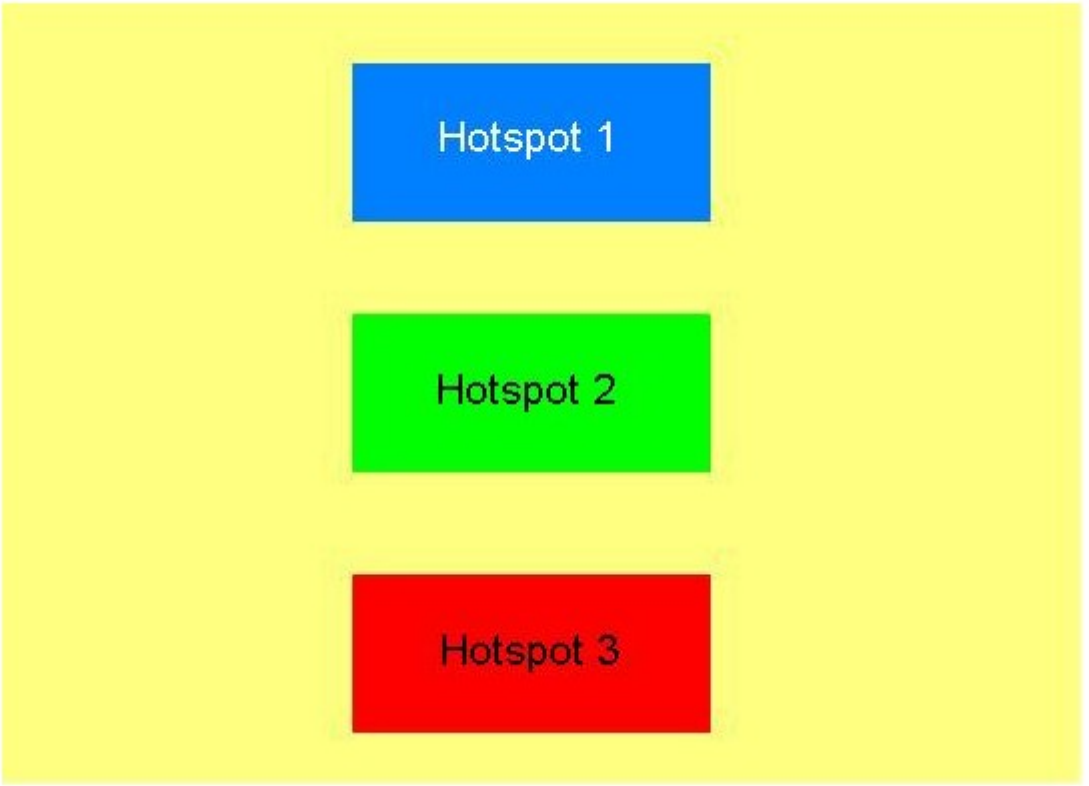
MultiView with 3 Views

View 3

Now showing View #3

<	January 2007					>
Sun	Mon	Tue	Wed	Thu	Fri	Sat
31	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	9	10

ImageMap Class Mixed HotSpotMode Example



Wątki

- System.Object
  - System.Threading.Thread
- public sealed class Thread
- Konstruktor:
  - `public Thread(ThreadStart start);`
- Własności:
  - `public static Thread CurrentThread {get;}`
  - `public bool IsAlive {get;}`
  - `public bool IsThreadPoolThread {get;}`
  - `public string Name {get; set;}`
  - `public ThreadPriority Priority {get; set;}`
  - `public ThreadState ThreadState {get;}`

Wątki

- Metody:
  - `public void Abort();`
  - `public bool Join(int millisecondsTimeout);`
  - `public void Resume();`
  - `public static void Sleep(int millisecondsTimeout);`
  - `public void Start();`
  - `public void Suspend();`
- Delegacje:
  - `public delegate void ThreadStart();`

```
using System;
using System.Threading;

public class ServerClass{
    public void InstanceMethod(){
        Console.WriteLine("ServerClass.InstanceMethod is running
            on another thread.");
        Thread.Sleep(3000);
        Console.WriteLine("The instance method called by the worker
            thread has ended.");
    }

    public static void StaticMethod(){
        Console.WriteLine("ServerClass.StaticMethod is running
            on another thread.");
        Thread.Sleep(5000);
        Console.WriteLine("The static method called by the worker
         thread has ended.");
    }
}

public class Simple{
    public static int Main(String[] args){
        Console.WriteLine("Thread Simple Sample");
        ServerClass serverObject = new ServerClass();
        Thread InstanceCaller = new Thread(new ThreadStart(
            serverObject.InstanceMethod));
        InstanceCaller.Start();
        Console.WriteLine("The Main() thread calls this after
            starting the new InstanceCaller thread.");
        Thread StaticCaller = new Thread(new ThreadStart(
            ServerClass.StaticMethod));
        StaticCaller.Start();
        Console.WriteLine("The Main() thread calls this after
            starting the new StaticCaller threads.");

        return 0;
    }
}
```

```
public class Work
{
    public static void Main()
    {
        Thread newThread = new Thread(
            new ParameterizedThreadStart(Work.DoWork));

        newThread.Start(42);

        Work w = new Work();
        newThread = new Thread(
            new ParameterizedThreadStart(w.DoMoreWork));

        newThread.Start("The answer.");
    }

    public static void DoWork(object data)
    {
        Console.WriteLine("Static thread procedure. Data='{0}'",
            data);
    }

    public void DoMoreWork(object data)
    {
        Console.WriteLine("Instance thread procedure. Data='{0}'",
            data);
    }
}


Static thread procedure. Data='42'
Instance thread procedure. Data='The answer'
```

```
using System;
using System.Threading;

public class ThreadWithState {
    private string boilerplate;
    private int value;

    public ThreadWithState(string text, int number) {
        boilerplate = text;
        value = number;
    }

    public void ThreadProc() {
        Console.WriteLine(boilerplate, value);
    }
}

public class Example {
    public static void Main() {
        ThreadWithState tws =
            new ThreadWithState("This report displays the number
             {0}.", 42);
        Thread t = new Thread(new ThreadStart(tws.ThreadProc));
        t.Start();
        Console.WriteLine("Main thread does some work, then waits.");
        t.Join();
        Console.WriteLine("Independent task has completed; main
          thread ends.");
    }
}
```

```
using System;
using System.Threading;

public class ThreadWithState {
    private string boilerplate;
    private int value;
    private ExampleCallback callback;

    public ThreadWithState(string text, int number,
                           ExampleCallback callbackDelegate)
    {
        boilerplate = text;
        value = number;
        callback = callbackDelegate;
    }
    public void ThreadProc() {
        Console.WriteLine(boilerplate, value);
        if (callback != null)
            callback(1);
    }
}

public delegate void ExampleCallback(int lineCount);

public class Example {
    public static void Main() {
        ThreadWithState tws = new ThreadWithState(
            "This report displays the number {0}.", 42,
            new ExampleCallback(ResultCallback));
        Thread t = new Thread(new ThreadStart(tws.ThreadProc));
        t.Start();
        Console.WriteLine("Main thread does some work, then waits.");
        t.Join();
        Console.WriteLine("Independent task has completed; main thread
          ends.");
    }
    public static void ResultCallback(int lineCount) {
        Console.WriteLine("Independent task printed {0} lines.",
          lineCount);
    }
}
```

Synchronizacja

- System.Object
  - System.Threading.Monitor
- public sealed class Monitor
- Metody:
  - `public static void Enter(object obj);`
  - `public static void Exit(object obj);`
  - `public static void Pulse(object obj);`
  - `public static bool TryEnter(object obj, int millisecondsTimeout);`
  - `public static bool Wait(object obj, int millisecondsTimeout);`

```
using System;
using System.Threading;
using System.Collections;
namespace MonitorCS1
{
    class MonitorSample
    {
        const int MAX_LOOP_TIME = 1000;
        Queue    m_smplQueue;
        public MonitorSample()
        {
            m_smplQueue = new Queue();
        }
        public void FirstThread()
        {
            int counter = 0;
            lock(m_smplQueue)
            {
                while(counter < MAX_LOOP_TIME)
                {
                    Monitor.Wait(m_smplQueue);
                    m_smplQueue.Enqueue(counter);
                    Monitor.Pulse(m_smplQueue);
                    counter++;
                }
            }
        }
        public void SecondThread()
        {
            lock(m_smplQueue)
            {
                Monitor.Pulse(m_smplQueue);
                while(Monitor.Wait(m_smplQueue,1000))
                {
                    int counter = (int)m_smplQueue.Dequeue();
                    Console.WriteLine(counter.ToString());
                    Monitor.Pulse(m_smplQueue);
                }
            }
        }
```

```
        //Return the number of queue elements.
        public int GetQueueCount()
        {
            return m_smplQueue.Count;
        }

        static void Main(string[] args)
        {
            //Create the MonitorSample object.
            MonitorSample test = new MonitorSample();
            //Create the first thread.
            Thread tFirst = new Thread(new ThreadStart(
                test.FirstThread));
            //Create the second thread.
            Thread tSecond = new Thread(new ThreadStart(
                test.SecondThread));
            //Start threads.
            tFirst.Start();
            tSecond.Start();
            //wait to the end of the two threads
            tFirst.Join();
            tSecond.Join();
            //Print the number of queue elements.
            Console.WriteLine("Queue Count = " +
             test.GetQueueCount().ToString());
        }
    }
}
```

```
using System;
using System.Collections;
using System.Threading;

namespace MonitorCS2
{
    class MonitorSample
    {
        private Queue m_inputQueue;

        public MonitorSample()
        {
            m_inputQueue = new Queue();
        }

        public void AddElement(object qValue)
        {
            //Lock the queue.
            Monitor.Enter(m_inputQueue);
            //Add element
            m_inputQueue.Enqueue(qValue);
            //Unlock the queue.
            Monitor.Exit(m_inputQueue);
        }

        public bool AddElementWithoutWait(object qValue)
        {
            //Determine whether the queue is locked
            if(!Monitor.TryEnter(m_inputQueue))
                return false;
            m_inputQueue.Enqueue(qValue);

            Monitor.Exit(m_inputQueue);
            return true;
        }
```

```
public bool WaitToAddElement(object qValue, int waitTime)
{
    //Wait while the queue is locked.
    if(!Monitor.TryEnter(m_inputQueue,waitTime))
        return false;
    m_inputQueue.Enqueue(qValue);
    Monitor.Exit(m_inputQueue);

    return true;
}

public void DeleteElement(object qValue)
{
    Monitor.Enter(m_inputQueue);
    int counter = m_inputQueue.Count;
    while(counter > 0)
    {
        //Check each element.
        object elm = m_inputQueue.Dequeue();
        if(!elm.Equals(qValue))
        {
            m_inputQueue.Enqueue(elm);
        }
        --counter;
    }
    Monitor.Exit(m_inputQueue);
}

public void PrintAllElements()
{
    Monitor.Enter(m_inputQueue);
    IEnumerator elmEnum = m_inputQueue.GetEnumerator();
    while(elmEnum.MoveNext())
    {
        Console.WriteLine(elmEnum.Current.ToString());
    }
    Monitor.Exit(m_inputQueue);
}
```

```
        static void Main(string[] args)
        {
            MonitorSample sample = new MonitorSample();

            for(int i = 0; i < 30; i++)
                sample.AddElement(i);
            sample.PrintAllElements();
            sample.DeleteElement(0);
            sample.DeleteElement(10);
            sample.DeleteElement(20);
            sample.PrintAllElements();

        }
    }
}
```

Synchronizacja 2

- System.Object
- System.MarshalByRefObject
- System.Threading.WaitHandle
- System.Threading.Mutex
- Konstruktor:
  - `public Mutex();`
- Metody:
  - `public void ReleaseMutex();`
  - `public static bool WaitAll(WaitHandle[] waitHandles);`
  - `public static int WaitAny(WaitHandle[] waitHandles);`
  - `public virtual bool WaitOne();`

```
using System;
using System.Threading;

class Test
{
    private static Mutex mut = new Mutex();
    private const int numIterations = 1;
    private const int numThreads = 3;

    static void Main()
    {
        for(int i = 0; i < numThreads; i++)
        {
            Thread myThread = new Thread(new ThreadStart(MyThreadProc));
            myThread.Name = String.Format("Thread{0}", i + 1);
            myThread.Start();
        }
    }
    private static void MyThreadProc()
    {
        for(int i = 0; i < numIterations; i++)
        {
            UseResource();
        }
    }

    private static void UseResource()
    {
        mut.WaitOne();

        Console.WriteLine("{0} has entered the protected area",
            Thread.CurrentThread.Name);

        Thread.Sleep(500);

        Console.WriteLine("{0} is leaving the protected area\r\n",
            Thread.CurrentThread.Name);

        mut.ReleaseMutex();
    }
}
```

ThreadPool

- Metody:
  - `public static void GetAvailableThreads(out int workerThreads, out int completionPortThreads);`
  - `public static void GetMaxThreads(out int workerThreads, out int completionPortThreads);`
  - `public static bool QueueUserWorkItem(WaitCallback callBack, object state);`
  - `public static RegisteredWaitHandle RegisterWaitForSingleObject(WaitHandle waitObject, WaitOrTimerCallback callBack, object state, int millisecondsTimeOutInterval, bool executeOnlyOnce);`
- Delegacje:
  - `[Serializable] public delegate void WaitCallback(object state);`

```
using System;
using System.Threading;
public class Example {
    public static void Main() {
        // Queue the task.
        ThreadPool.QueueUserWorkItem(new WaitCallback(ThreadProc));

        Console.WriteLine("Main thread does some work, then sleeps.");
        // If you comment out the Sleep, the main thread exits before
        // the thread pool task runs.  The thread pool uses background
        // threads, which do not keep the application running.  (This
        // is a simple example of a race condition.)
        Thread.Sleep(1000);

        Console.WriteLine("Main thread exits.");
    }

    // This thread procedure performs the task.
    static void ThreadProc(Object stateInfo) {
        // No state object was passed to QueueUserWorkItem, so
        // stateInfo is null.
        Console.WriteLine("Hello from the thread pool.");
    }
}
```

```
using System;
using System.Threading;

public class TaskInfo {
    public RegisteredWaitHandle Handle = null;
    public string OtherInfo = "default";
}

public class Example {
    public static void Main(string[] args) {
        // The main thread uses AutoResetEvent to signal the
        // registered wait handle, which executes the callback
        // method.
        AutoResetEvent ev = new AutoResetEvent(false);

        TaskInfo ti = new TaskInfo();
        ti.OtherInfo = "First task";
        // The TaskInfo for the task includes the registered wait
        // handle returned by RegisterWaitForSingleObject.  This
        // allows the wait to be terminated when the object has
        // been signaled once (see WaitProc).
        ti.Handle = ThreadPool.RegisterWaitForSingleObject(
            ev, new WaitOrTimerCallback(WaitProc), ti, 1000, false
        );

        // The main thread waits three seconds, to demonstrate the
        // time-outs on the queued thread, and then signals.
        Thread.Sleep(3100);
        Console.WriteLine("Main thread signals.");
        ev.Set();

        // The main thread sleeps, which should give the callback
        // method time to execute.  If you comment out this line, the
        // program usually ends before the ThreadPool thread can execute.
        Thread.Sleep(1000);
        // If you start a thread yourself, you can wait for it to end
        // by calling Thread.Join.  This option is not available with
        // thread pool threads.
    }
```

```
  // The callback method executes when the registered wait times out,
    // or when the WaitHandle (in this case AutoResetEvent) is signaled.
    // WaitProc unregisters the WaitHandle the first time the event is
    // signaled.
    public static void WaitProc(object state, bool timedOut) {
        // The state object must be cast to the correct type, because
        // the signature of the WaitOrTimerCallback delegate specifies
        // type Object.
        TaskInfo ti = (TaskInfo) state;

        string cause = "TIMED OUT";
        if (!timedOut) {
            cause = "SIGNALED";
            // If the callback method executes because the WaitHandle is
            // signaled, stop future execution of the callback method
            // by unregistering the WaitHandle.
            if (ti.Handle != null)
                ti.Handle.Unregister(null);
        }

        Console.WriteLine("WaitProc( {0} ) executes on thread {1};
             cause = {2}.", ti.OtherInfo,
            Thread.CurrentThread.GetHashCode().ToString(),
            cause
        );
    }
}
```

Timer

- System.Object
  - System.MarshalByRefObject
    - System.Threading.Timer
- public sealed class Timer : MarshalByRefObject, IDisposable
- Metody:
  - public Timer(TimerCallback callback, object state, int dueTime, int period);
  - public bool Change(int dueTime, int period);
- Delegacje:
  - [Serializable]
    public delegate void TimerCallback(object state);

```
using System;
using System.Threading;

class TimerExampleState
{
   public int counter = 0;
   public Timer tmr;
}

class App
{
   public static void Main()
   {
      TimerExampleState s = new TimerExampleState();
      TimerCallback timerDelegate = new TimerCallback(CheckStatus);
      Timer timer = new Timer(timerDelegate, s,1000, 1000);
      s.tmr = timer;
      while(s.tmr != null)
         Thread.Sleep(0);
      Console.WriteLine("Timer example done.");
   }

   static void CheckStatus(Object state)
   {
      TimerExampleState s =(TimerExampleState)state;
      s.counter++;
      Console.WriteLine("{0} Checking Status {1}.",
        DateTime.Now.TimeOfDay, s.counter);
      if(s.counter == 5)
      {
        (s.tmr).Change(10000,100);
        Console.WriteLine("changed...");
      }
      if(s.counter == 10)
      {
        Console.WriteLine("disposing of timer...");
        s.tmr.Dispose();
        s.tmr = null;
      }
   }
}
```