ASP.NET Web Parts

Krzysztof Jeliński

Dawid Gawroński

Toruń 2011/2012

ASP.NET Web Parts

ASP.NET Web Parts jest zintegrowanym zbiorem kontrolek do tworzenia stron Web, które pozwalają użytkownikom końcowym modyfikować zawartość, wygląd i zachowanie stron z poziomu przeglądarki. Modyfikacje mogą być zastosowane dla wszystkich użytkowników strony, bądź też indywidualnie. Gdy użytkownicy modyfikują strony i kontrolki, zmiany te mogą zostać zapisane w celu zapamiętania preferencji użytkownika dla przyszłych sesji przeglądarki. Te funkcje Web Parts oznaczają, że deweloperzy mogą upoważnić użytkowników końcowych do dynamicznej personalizacji aplikacji webowej, bez ingerencji dewelopera bądź administratora.

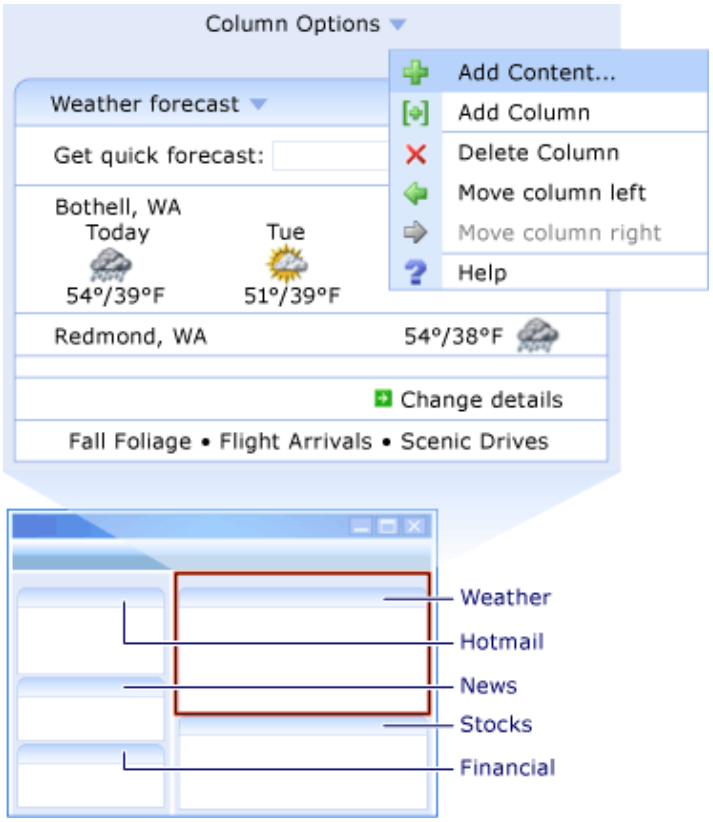
Web Parts pozwala użytkownikowi końcowemu na personalizacje strony według własnych potrzeb bądź gustu. Stworzoną z wykorzystaniem Web Parts aplikację internetową można porównać do klocków, które układamy według własnego *widzi mi się*. Każdy z takich klocków pełni odrębną funkcjonalność. Ponadto to użytkownik określa które elementy mają być wyświetlane i w jaki sposób. Jest to bardzo wygodne rozwiązanie ponieważ każdy użytkownik ma taki widok jaki on sam preferuje.

Używanie zbioru kontrolek Web Parts przez dewelopera pozwala mu na zezwolenie użytkownikom końcowym na:

- **Personalizacje zawartości strony**. Użytkownicy mogą dodawać na stronie nowe komponenty, usuwać je, ukrywać lub minimalizować jak zwyczajne okna.
- **Personalizację wyglądu strony**. Użytkownicy mogą przeciągać kontrolki WebPart do różnych obszarów na stronie. Mogą również zmieniać ich wygląd, właściwości i zachowanie.
- **Eksportować i importować kontrolki**. Użytkownicy mogą importować bądź eksportować ustawienia (właściwości, wygląd a nawet zawarte w nich dane) kontrolek Web Part, by używać ich na innych stronach. Zmniejsza to ilość wpisów danych i czas konfiguracji.
- **Tworzenie połączeń**. Użytkownicy mogą tworzyć połączenia pomiędzy kontrolkami, tak aby na przykład wyświetlić wykres dla danych zawartych w innej kontrolce. Użytkownicy mogą nie tylko tworzyć połączenia, ale również ustawiać ich wygląd i sposób w jaki wykres będzie prezentowany.
- **Zarządzać i personalizować ustawienia strony**. Autoryzowani użytkownicy mogą konfigurować ustawienia strony i określić kto ma mieć do niej dostęp, ustawiać grupowe prawda dostępu itp. Np. Użytkownik z uprawnieniami administratora może ustawić kontrolkę aby była współdzielona przez wszystkich użytkowników, lecz zabronić zmieniania jej ustawień przez użytkowników nie mających praw administracyjnych.

Zbiór kontrolek Web Parts dzieli się na trzy główne bloki: personalizację, strukturalne komponenty interfejsu użytkownika (UI) i właściwych kontrolek UI Web Parts. Spora część pracy na rzecz rozwoju koncentruje się głównie na kontrolkach Web Part, które są prostymi kontrolkami ASP.NET z możliwością korzystania z funkcjonalności Web Parts.

Przeanalizuj przykład z rysunku 1, aby zobaczyć jak kontrolki Web Parts mogą być użyte do zbudowania spersonalizowanej strony Web.



Rysunek 1: Typowa strona Web Parts

Strona ta zawiera kilka podstawowych elementów aplikacji Web Parts.

- Obszary. Obszary mają określony rozmiar i lokalizację na stronie. Na rysunku są dwie kolumny, które mogą zawierać kontrolki: jedna posiada kontrolki Weather i Stock, w drugiej znajdują się kontrolki Hotmail i News. Kolumny te w terminologii Web Parts są nazywane obszarami-miejscami na stronie które zawierają kontrolki. Obszary istnieją by móc rozmieszczać kontrolki na stronie i żeby zapewnić im interfejs użytkownika. Na stronie może istnieć jeden lub wiele obszarów, z których każdy może zawierać jeden lub wiele kontrolek Web Parts. Każdy obszar ma albo pionową albo poziomą orientację na stronie. Obszary w których mogą pojawiać się komponenty to WebPartZone.
- Kontrolki Web Parts wewnątrz obszarów. Każda kontrolka ma akcję, którą może wykonać użytkownik i która może objawiać się jako link, przycisk lub klikalny obrazek na kontrolce. Zauważ, że na rysunku 1 każda kontrolka ma przycisk znajdujący się w pasku tytułu, który rozwija menu typu drop-down. W menu dla każdej kontrolki są opcje pozwalające zmienić dane określone dla tej kontrolki i inne opcje w celu przeprowadzenia wspólnych działań takich jak przenoszenie kontrolki, usuwanie kontrolki lub uzyskiwanie pomocy. Niektóre kontrolki jak kontrolka *Weather*, pozwalają użytkownikom na ich spersonalizowanie w wyniku czego kontrolki wyświetlają tylko dane dotyczące konkretnego użytkownika.
- Linki umożliwiające rozszerzoną personalizację. Pozwalają one użytkownikom na zmianę zawartości strony, koloru i layoutu strony. Na przykład, jeśli użytkownicy klikną na „Add Column”, aplikacja Web Parts pozwoli im na dodanie kolejnej kolumny na stronie. Gdy użytkownicy klikną na „Add Content”, zostanie wyświetlona lista kontrolek, które mogą zostać opcjonalnie dodane na stronę. Przykładem może być kontrolka wykresów akcji. Użytkownik może dodać tą kontrolkę do jednego z obszarów na stronie, a następnie połączyć ją z istniejącą kontrolką zawierającą notowania giełdowe aby zrobić wykres tych danych.

Web Parts udostępnia kilka trybów wyświetlania strony, są to:

- BrowseDisplayMode – Tryb przeglądania strony. Edycja kontrolek Web Parts nie jest możliwa.
- CatalogDisplayMode – Pokazuje okno katalogu, za pomocą którego dodajemy, bądź przywracamy na stronę kontrolki Web Parts.
- ConnectDisplayMode – W tym trybie określamy połączenia między kontrolkami Web Parts.
- DesignDisplayMode – Tryb służący do zmiany położenia kontrolek Web Parts na stronie.
- EditDisplayMode – Jest to tryb służący do edytowania ustawień kontrolek Web Parts.

Dostępność niektórych trybów wyświetlania jest zależna od tego jakie strefy zostały dodane do strony (CatalogDisplayMode wymaga CatalogZone, EditDisplayMode wymaga EditorZone).

Każda strona aplikacji internetowej wykorzystująca komponenty Web Parts potrzebuje kontrolki WebPartManager do poprawnego działania. Dlatego podczas tworzenia strony, korzystającej z Web Parts dodanie tej kontrolki to pierwsza czynność jaką powinniśmy wykonać. Kontrolka ta w Visual Studio nie ma reprezentacji graficznej, ponieważ jest ona niewidoczna dla użytkownika i służy jedynie zarządzaniu zawartymi na stronie komponentami Web Parts.

Web Parts udostępnia funkcjonalność zwaną katalogiem, która jest swego rodzaju schowkiem na kontrolki. Za jego pomocą można dodawać na stronę nowe kontrolki bądź przywracać zamknięte. Katalog pozwala na rozróżnienie komponentów związanych ze stroną jak i tych związanych z całym serwisem. Ponadto przy jego użyciu realizuje się importowanie spersonalizowanych przez użytkownika komponentów. Spersonalizowane komponenty są reprezentowane przez pliki xml, które dobrze służą do opisu struktur danych. Eksportowanie komponentów odbywa się w prosty sposób, ponieważ Web Parts udostępnia odpowiednie narzędzia. Wszystko sprowadza się do ustawienia odpowiednich opcji w pliku konfiguracyjnym. Niektóre komponenty mogą przechowywać poufne dane, których użytkownik mógłby nie chcieć ujawniać innym. Dlatego też istnieje możliwość eksportu tylko podstawowych ustawień opisujących komponent, tak by cenne dane nie wyciekły w niepowołane ręce. Przykład zastosowania importowania i eksportowania ustawień komponentów jest przedstawiony w tutorialu.

Wygodnym mechanizmem w Web Parts jest EditorPart. Gdy umieścimy go na stronie (a dokładniej w EditorZone) i użytkownik przejdzie w tryb edycji strony, dopiero wtedy EditorPart stanie się widoczny. Służy on do zmiany ustawień, wyglądu i zachowania komponentów na stronie.

Wybrane właściwości kontrolek Web Part wraz z krótkim opisem:

- AllowClose – Określa czy komponent może zostać zamknięty.
- AllowEdit – Określa czy ustawienia komponentu mogą być zmieniane.
- AllowZoneChange – Określa czy komponent może zostać przeniesiony na inny WebPartZone.
- CssClass – Nazwa klasy komponentu. Przydatne w stylowaniu strony przy użyciu CSS.
- Title – Tytuł który jest wyświetlany na pasku komponentu.
- ExportMode – Określa sposób eksportu dla komponentu.

Wykorzystanie dobrodziejstw Web Parts oraz .NET daje naprawdę ogromne możliwości. Web Part jest doskonałym narzędziem do tworzenia dynamicznych i w pełni konfigurowalnych aplikacji internetowych. W celu uzyskania większej ilości informacji, odsyłamy na oficjalną stronę MSDN Microsoftu, gdzie znajduje się szczegółowa dokumentacja.

Tworzenie przykladowej strony korzystajacej z Web Parts krok po kroku:

I. Tworzymy stronę zawierającą kontrolki web parts

1. Tworzymy nowy pusty projekt strony (**File** > **New** > **Web Site** > **ASP .NET Empty Web Site**) i dodajemy nową pustą stronę do projektu (Prawy przycisk myszy na utworzonym projekcie > **Add New Item …** > zaznaczamy **Web Form** i klikamy **Add**). Nazywamy ją domyślnie, czyli „Default.aspx".
2. Przełączamy się na widok **Design**. W **Toolbox** rozwijamy zakładkę **WebParts** i przeciągamy z niej na stronę kontrolkę **WebPartManager**. Dodajemy ją na samej górze strony, tak by znajdowała się przed blokiem div. Kontrolka ta jest niewidoczna dla przeglądarki a w Visual Studio widziana jest jako szary prostokąt.
3. Klikamy w bloku div. Na pasku narzędzi w **Block Format** wybieramy **Heading 1** i wpisujemy „Strona korzystająca z dobrodziejstw Web Parts". Następnie z **toolbox** rozwijamy **HTML** i przeciągamy element **div** za nowo utworzony **h1**. W ten sposób umieszczamy element **div** w istniejącym divie.
4. Klikamy w zagnieżdżonym divie a następnie, w menu **Table** klikamy **Insert Table.** W polu **row** wpisujemy 1, natomiast w polu **columns** wpisujemy 3. Ustawiamy Layout:**Alignment** na **Center** i Background:**Color** na **#CCFFCC** a następnie klikamy **OK**.
5. Z **Toolbox**>**WebParts** przeciągamy **WebPartZone** do lewej kolumny. Czynność powtarzamy dla środkowej kolumny.
6. Klikamy prawym przyciskiem myszy na kontrolce WebPartZone w lewej kolumnie i klikamy **Properties**. Ustawiamy **ID** na „LewyObszar" i **HeaderText** na „Lewy obszar". Czynność powtarzamy dla WebPartZone w środkowej kolumnie i ustawiamy dla niego **ID** na „GlownyObszar" oraz **HeaderText** na „Główny obszar".
7. W **Toolbox** rozwijamy zakładkę **Standard** i przeciągamy **Label** do zawartości obszaru głównego.
8. Przełącz się na widok **Source**. Zauważ, że element **Label** został zawarty w **ZoneTemplate** obszaru GlownyObszar. Dodaj atrybut **title** dla elementu **label** i ustaw jego wartość na „Zawartość" oraz usuń istniejący atrybut **Text**. W kontrolce label dodaj tekst np. „<h2>Witaj na mojej stronie domowej</h2>". Następnie zapisz dokonane zmiany w pliku Default.aspx.

```
<asp:WebPartZone ID="GlownyObszar" runat="server" HeaderText="Główny obszar">
   <ZoneTemplate>
      <asp:Label ID="Label1" runat="server" title="Zawartość">
         <h2>Witaj na mojej stronie domowej</h2>
      </asp:Label>
   </ZoneTemplate>
</asp:WebPartZone>
```

II. Tworzenie własnej kontrolki

1. Klikamy prawym przyciskiem myszy na projekcie w Solution Explorerze, klikamy **Add New Item ...** i wybieramy **Web User Control**. Jako nazwę wpisujemy „Google.ascx". Upewniamy się, że **Place code in separate file** jest zaznaczone i klikamy **Add**.
2. Pobieramy z internetu logo googla (najlepiej z przeźroczystością). W Solution Explorerze tworzymy nowy katalog dla projektu, w którym będziemy przechowywać grafikę. Klikamy prawym-myszy na projekcie i wybieramy **New Folder**. Nazywamy go „Obrazy". Następnie prawy-myszy na nowoutworzonym folderze Obrazy i **Add Existing Item**. Wskazujemy pobrany plik zawierający logo wyszukiwarki Google.
3. Przełączamy się na widok **Design**. Przeciągamy do Google.ascx element **div** a następnie

przeciągamy do jego zawartości dwa nowe divy. Do pierwszego z zagnieżdżonych divów przeciągamy obrazek z logo Google. Będąc w drugim z zagnieżdżonych divów klikamy na menu **Table**, wybieramy **Insert Table** i dodajemy tabelę 1x2. Wo lewej kolumny dodajemy element **TextBox** a do prawej **Button**. Zmieniamy tekst wyświetlany na Buttonie na „Szukaj”. Źródło Google.ascx powinna prezentować się w następujący sposób:

```
<%@ Control Language="C#" AutoEventWireup="true" CodeFile="Google.ascx.cs" Inherits="Google"
%>
<div>
   <div>
      <img alt="Google" class="style3" src="Obrazy/GoogleLOGO.png"/>
   </div>

   <div>
      <table>
         <tr>
            <td>
               <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
            </td>
            <td>
               <asp:Button ID="Button1" runat="server" Text="Szukaj" />
            </td>
         </tr>
      </table>
   </div>
</div>
```

4. Następnie w widoku **Design** klikamy dwukrotnie na Button. Utworzy się w ten sposób metoda obsługująca kliknięcie przycisku Szukaj. Zawartość całego pliku Google.ascx.cs:

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

public partial class Google : System.Web.UI.UserControl
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        Response.Write(Page.IsValid);
        string zapytanie = HttpUtility.UrlEncode(TextBox1.Text);
        Response.Redirect("http://www.google.com/search?q=" + zapytanie);
    }
}
```

5. Tworzymy jeszcze jeden **Web User Control** i nazywamy go „Kalendarz”. Otwórz plik Kalendarz.ascx i ustaw tryb widoku na **Design**. Następnie z **Toolboxa** rozwijamy zakładkę **Standard** i przeciągamy element **Calendar** na formę. Klikamy na **autoformat** i wybieramy schemat **Colorful 1**. Zapisujemy zmiany w pliku Kalendarz.ascx i zamykamy go.

III. Dodajemy kontrolki Web Parts do Lewego Obszaru

1. Otwieramy plik Default.aspx i przechodzimy do trybu **Design**.
2. Przeciągamy Kalendarz.ascx z Solution Explorera do LewyObszar. Przełączamy się w tryb Source i dodajemy atrybut **title** o wartości „Kalendarz”.

3. Będąc ciągle w widoku **Source** szukamy elementu **asp:WebPartZone** dla LewyObszar. Dodajemy do jego zawartości **Label** w którym będą odnośniki do stron www. Dodatkowo dodajemy atrybut **title** dla elementu **label** i ustawiamy jego wartośc na „Odnośniki”.

```
<asp:WebPartZone ID="LewyObszar" runat="server" HeaderText="Lewy obszar">
   <ZoneTemplate>
      <asp:Label runat="server" id="Odnosniki" title="Odnośniki">
         <a href="http://www.fizyka.umk.pl">WFAiIS UMK</a><br />
         <a href="http://www.pajacyk.pl">Pajacyk</a><br />
         <a href="http://www.torun.pl">Torun.pl</a><br />
      </asp:Label>
      <uc1:Kalendarz ID="Kalendarz1" runat="server" title="Kalendarz" />
   </ZoneTemplate>
</asp:WebPartZone>
```

IV. Pozwolenie użytkownikom na edycje i zmienianie układu strony.

1. Dodajemy nowy Web User Control do projektu i nazywamy go MenuTrybuWyswietlania.ascx. Upewniamy się, że opcja **Place source code in separate file** jest odznaczona i klikamy **Add**.
2. Przełączamy się w tryb wyświetlania Source i zamieniamy całą zawartość pliku MenuTrybuWyswietlania.ascx zamieniamy na:

```
<%@ control language="C#" classname="DisplayModeMenuCS"%>
<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);
      }
    }

    if (_manager.Personalization.CanEnterSharedScope)
    {
      Panel2.Visible = true;
      if (_manager.Personalization.Scope ==
        PersonalizationScope.User)
        RadioButton1.Checked = true;
      else
        RadioButton2.Checked = true;
    }
  }

  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)
  {
    ListItemCollection items = DisplayModeDropdown.Items;
    int selectedIndex =
      items.IndexOf(items.FindByText(_manager.DisplayMode.Name));
    DisplayModeDropdown.SelectedIndex = selectedIndex;
  }

  protected void LinkButton1_Click(object sender, EventArgs e)
  {
    _manager.Personalization.ResetPersonalizationState();
  }

  protected void RadioButton1_CheckedChanged(object sender, EventArgs e)
  {
    if (_manager.Personalization.Scope ==
      PersonalizationScope.Shared)
      _manager.Personalization.ToggleScope();
  }

  protected void RadioButton2_CheckedChanged(object sender,
    EventArgs e)
  {
    if (_manager.Personalization.CanEnterSharedScope &&
        _manager.Personalization.Scope ==
          PersonalizationScope.User)
        _manager.Personalization.ToggleScope();
  }
</script>
<div>
  <asp:Panel ID="Panel1" runat="server"
    Borderwidth="1"
    Width="230"
    BackColor="lightgray"
    Font-Names="Verdana, Arial, Sans Serif" >
    <asp:Label ID="Label1" runat="server"
      Text=" Tryb Wyświetlania"
      Font-Bold="true"
      Font-Size="8"
      Width="120" />
    <asp:DropDownList ID="DisplayModeDropdown" runat="server"
      AutoPostBack="true"
      Width="120"
      OnSelectedIndexChanged="DisplayModeDropdown_SelectedIndexChanged" />
    <asp:LinkButton ID="LinkButton1" runat="server"
      Text="Resetuj"
      ToolTip="Resetuje zapamietane preferencje uzytkownika odnosnie strony."
      Font-Size="8"
      OnClick="LinkButton1_Click" />
    <asp:Panel ID="Panel2" runat="server"
      GroupingText="Personalization Scope"
      Font-Bold="true"
      Font-Size="8"
      Visible="false" >
      <asp:RadioButton ID="RadioButton1" runat="server"
        Text="Użytkownik"
        AutoPostBack="true"
        GroupName="Scope"
        OnCheckedChanged="RadioButton1_CheckedChanged" />
      <asp:RadioButton ID="RadioButton2" runat="server"
        Text="Współdzielone"
```

```
        AutoPostBack="true"
        GroupName="Scope"
        OnCheckedChanged="RadioButton2_CheckedChanged" />
    </asp:Panel>
  </asp:Panel>
</div>
```

3. Zapisz plik i przełącz się do trybu **Design**. Otwórz plik Default.aspx i przeciągnij MenuTrybuWyswietlania.ascx na element h1 zawierający tekst „Strona korzystająca z dobrodziejstw Web Parts”.
4. Z zakładki **WebParts** z **Toolbox** przeciągnij kontrolkę **EditorZone** na prawą (pustą) kolumnę.
5. Z zakładki **WebParts** z **Toolbox** przeciągnij kontrolki **ApperanceEditorPart** i **LayoutEditorPart** na kontrolkę **EditorZone**.
6. Zawartość prawej kolumny z pliku Default.aspx powinna zawierać następujący kod:

```
<td>
   <asp:EditorZone ID="EditorZone1" runat="server">
      <ZoneTemplate>
         <asp:AppearanceEditorPart ID="AppearanceEditorPart1" runat="server" />
         <asp:LayoutEditorPart ID="LayoutEditorPart1" runat="server" />
      </ZoneTemplate>
   </asp:EditorZone>
</td>
```

V. Dodawanie Web Parts „w biegu”.

1. Otwórz plik Default.aspx i przełącz się w tryb widoku **Source**. Z zakładki **WebParts** w **Toolbox**, przeciągnij kontrolkę **CatalogZone** do prawej kolumny tuż pod **EditorZone**. Obie kontrolki mogą znajdować się w tej samej kolumnie, ponieważ nie będą one nigdy wyświetlane równocześnie.
2. W oknie **Properties** ustaw wartość parametru **HeaderText** na „Add Web Parts” dla elementu **CatalogZone**.
3. Przełącz się w tryb **Design**. Z zakładki WebParts w Toolbox, przeciągnij **DeclarativeCatalogPart** do zawartości **CatalogZone**. Kliknij strzałkę w górnym prawym rogu i kliknij **Edit Templates**.
4. Następnie przeciągnij z Solution Explorera element Google.ascx do zawartości **WebPartsTemplate**. Następnie przełączamy się w tryb **Source** i dodajemy elementowi atrybut **title** o wartości „Wyszukiwarka Google”.

VI. Dodawanie uprzednio zamkniętych elementów Web Parts

1. Aby mieć możliwość ponownego dodania zamkniętego elementu musimy do zawartości **CatalogZone** przeciągnąć **PageCatalogPart** z zakładki **WebParts**

VII. Tworzenie połączeń pomiędzy elementami Web Parts.

1. Otwieramy plik Default.aspx w trybie **Design**. Dodajemy na jego dole nowy div i przeciągamy do niego **ConnectionZone** z zakładki **WebParts** w **Toolbox**.
2. Dodajemy do projektu nowy element typu Web User Control o nazwie Provider.aspx. Upewniamy się, że opcja Place code in separate file jest zaznaczona i klikamy **Add**.
3. Zawartość pliku Provider.aspx:

```
<%@ Control Language="C#" AutoEventWireup="true" CodeFile="Provider.ascx.cs"
Inherits="Provider" %>
<table width="100%" cellpadding="4" cellspacing="0" bgcolor="#ececec">
  <tr>
    <td align="left" valign="top">
      Ten element Web Part jest typu Provider. Służy on do przesyłania
      tekstu wpisanego do textboxa do elementu Web Part typu Consumer.
    </td>
  </tr>
  <tr>
    <td>
      <asp:TextBox ID="TextBox1" MaxLength="16" Runat="server" />
      <asp:Button ID="Button1" Text="Prześlij" Runat="server" />
    </td>
  </tr>
</table>
```

4. Zawartość pliku Provider.ascx.cs

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



public partial class Provider : System.Web.UI.UserControl, ITextTransfer
{


    [ConnectionProvider("Text", "TextProvider")]
    public ITextTransfer GetTextTransferInterface()
    {
        return this;
    }

    public string GetText()
    {
        return TextBox1.Text;
    }

    protected void Page_Load(object sender, EventArgs e)
    {

    }
}
```

5. Musimy jeszcze dodać interfejs ITextTransfer. Klikamy prawym-myszy na projekcie i wybieramy **Add New Item …** . Następnie wybieramy Class, wpisujemy nazwę ITextTransfer i klikamy **Add**. Zostaniemy zapytani czy zgadzamy się na utworzenie dodatkowego katalogu. Odpowiadamy twierdząco. Zawartość pliku ITextTransfer:

```
public interface ITextTransfer
{
    string GetText();
}
```

6. Dodajemy do projektu nowy element typu **Web User Control** o nazwie Consumer.aspx. Upewniamy się, że opcja **Place code in separate file** jest zaznaczona i klikamy **Add**.
7. Zawartość pliku Consumer.aspx

```
<%@ Control Language="C#" AutoEventWireup="true" CodeFile="Consumer.ascx.cs"
Inherits="Consumer" %>
<table width="100%" height="88pt" cellpadding="4" cellspacing="0"
  bgcolor="#ececec">
  <tr>
    <td align="left" valign="top">
      To jest element Web Part typu Consumer.
      Pozyskuje on tekst z elementu Web Part typu Provider.
    </td>
  </tr>
  <tr>
    <td>
      <asp:Label ID="Label1" Font-Size="14pt" Font-Bold="true" Runat="server" />
    </td>
  </tr>
</table>
```

8. Zawartość pliku Consumer.aspx.cs

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

public partial class Consumer : System.Web.UI.UserControl
{
    [ConnectionConsumer("Text", "TextConsumer")]
    public void GetTextTransferInterface(ITextTransfer provider)
    {
        Label1.Text = provider.GetText();
    }
    protected void Page_Load(object sender, EventArgs e)
    {

    }
}
```

9. Będąc w trybie widoku **Design** klikamy na strzałkę w prawym górnym rogu elementu „Katalog Elementów dodatkowych” i wybieramy **Edit Templates**. Przeciągnij elementy Provider.ascx oraz Consumer.aspx do zawartości **WebPartsTemplate**. Dzięki temu zabiegowi dajemy użytkownikowi możliwość dodania tych elementów na stronę. Należy również pamiętać o przełączeniu się w tryb **Source** i nadaniu elementom atrybutów **title** z odpowiednimi nazwami.

VIII. Testowanie tworzenia połączeń

1. Ustawiamy tryb wyświetlania na **Catalog**. Dodajemy na stronę elementy **Consumer** i **Provider**. Następnie ustawiamy tryb wyświetlania strony na **Connect** i klikamy małą strzałkę przy nazwie komponentu **Web Part** i wybieramy **connect**. Na dole strony pojawi się **Connections Zone** za pomocą którego tworzymy połączenie między elementami.

IX. Import i Eksport ustawień web parts.

1. Aby umożliwić importowanie spersonalizowanych elementów web part na stronę. Należy uprzednio dodać do **CatalogZone** w sekcji **ZoneTemplate** kontrolkę **ImportCatalogPart**. W tym celu wybieramy w **Toolbox** kontrolkę **ImportCatalogPart** i przeciągamy ją na **CatalogZone**. Teraz mamy już dodany element odpowiedzialny za importowanie.
2. Aby umożliwić eksport komponentów musimy w pliku konfiguracyjnym web.config dodać opowiednie parametry.

```
<?xml version="1.0"?>

<configuration>
  <system.web>
    <webParts enableExport="true"></webParts>
    <compilation debug="true" targetFramework="4.0"/>
  </system.web>
</configuration>
```

   Chodzi tu o dodanie parametru **enableExport** i ustawienie jego wartości na **true**.

3. Aby zezwolić użytkownikowi na eksportowanie konkretnego komponentu web part należy w pliku Default.aspx odszukać komponent który nas interesuje a następnie dodać mu parametr **ExportMode**. Parametr ten może przyjmować następujące wartości.
   a. All
   b. None (wartość domyślna niepozwalająca na eksport)
   c. NonSensitiveData (eksport ustawień za wyjątkiem danych prywatnych)

   W naszym tutorialu dodajemy możliwość eksportu komponentowi z wyszukiwarką Google. W tym celu odszukujemy odpowiedni wpis w pliku i zmieniamy jego zawartość na:

```
<uc3:Google ID="Google1" runat="server" ExportMode = "All" title="Wyszukiwarka Google" />
```

4. Teraz możemy przetestować, że wszystko działa jak należy. Startujemy stronę, dodajemy na nią komponent z wyszukiwarką Google. Następnie klikamy na strzałce w górnej prawej części i wybieramy opcję eksport. Następnie zapisujemy plik na dysku. Gdy mamy już wyeksportowany spersonalizowany komponent na dysku, musimy przejść do trybu **Catalog**. Teraz widzimy, że mamy w katalogu pozycję **Imported Web Part Catalog (0)**. Klikamy i wybieramy pobrany uprzednio plik z komponentem, a następnie zatwierdzamy przyciskiem **Upload**. Komponent zostanie dodany do strony, co będzie symbolizowane inkrementacją liczby znajdującej się w nawiasie (jest to licznik wczytanych komponentów). Następnie wybieramy **WebPartZone** do którego chcemy dodać komponent i cieszymy się dodaniem go na stronie.
5. Jeśli chcemy umożliwić użytkownikom eksport wszystkich komponentów, a nie tylko kilku wybranych, nieefektywnym rozwiązaniem byłoby przypisywanie każdemu z osobna parametru **ExportMode**. Problem ten można jednak łatwo rozwiązać z wykorzystaniem pętli typu **foreach**. W pliku Default.aspx.cs dodajemy dwie pętle, które dla każdego **WebPart**, zawartych w każdym **WebPartZone** ustawi odpowiedni parametr. Poniżej znajduje się kod z pliku Default.aspx.cs

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

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        foreach (WebPartZone wz in WebPartManager1.Zones)
        {
            foreach (WebPart wp in wz.WebParts)
            {
                wp.ExportMode = WebPartExportMode.NonSensitiveData;
            }
        }

    }
}
```

Proponujemy ustawienie trybu eksportu na **NonSensitiveData** aby zminimalizować prawdopodobieństwo wycieku poufnych informacji.

6. Jeśli teraz uruchomimy naszą stronę w przeglądarce widzimy, że każdy Web Part ma opcję eksportu.

X. Upiększanie

1. Na zakończenie dodamy jeszcze małą, lecz przydatną rzecz. Chodzi nam mianowicie o dodawanie ikonek w paskach tytułowych Web Part. W zależności od tego, gdzie komponent jest zdefiniowany, musimy to zrobić w różny sposób (dotyczy to wszystkich innych właściwości komponentów). Własność, która chcemy zmodyfikować to **TitleIconImageUrl**.
2. Komponent zdefiniowany w kodzie. Zmieńmy logo dla komponentu o **ID**=”Label1” wyświetlający domyślnie w obszarze głównym tekst: „Witaj na mojej stronie domowej”. Wystarczy bezpośrednio w kodzie dopisać parametr z określoną wartością.

```
<asp:Label ID="Label1" runat="server" title="Zawartość" TitleIconImageUrl="~/Obrazy/logo-
umk.gif">
```

3. W przypadku komponentu dodanego do projektu jako **Web User Control**, sprawa ma się nieco inaczej. Należy wówczas w metodzie **Page_Load** napisać:

```
protected void Page_Load(object sender, EventArgs e)
    {

        GenericWebPart gwp = (GenericWebPart)this.Parent;

        gwp.TitleIconImageUrl="~/Obrazy/google-ico.png";


    }
```

4. Według nas lepszym rozwiązaniem jest tworzenie nowych Web User Control. Uważamy tak, ponieważ wszystkie komponenty są widoczne w Solution Explorerze. Ponadto nie musimy przeszukiwać całego długiego kodu w pliku aspx. Jest to rozwiązanie o wiele bardziej eleganckie i nie wprowadza bałaganu w kodzie. Ponadto o wiele łatwiej jest nam korzystać z funkcjonalności .NET i języka C# (Inteli-sense jest bardzo przydatnym narzędziem).

Dziękujemy za ukończenie tego tutorialu :)

Źródła:

[1]. **http://msdn.microsoft.com/en-us/library/e0s9t4ck.aspx**

[2]. http://ondotnet.com/pub/a/dotnet/2005/05/23/webparts_1.html?page=2

[3]. http://windowsdevcenter.com/pub/a/windows/2006/01/10/what-are-web-parts.html?page=4

[4]. http://www.c-sharpcorner.com/uploadfile/a_anajwala/building_webparts.mht08042005042119am/building_webparts.mht.aspx

[5]. http://dotnetslackers.com/articles/aspnet/UsingWebPartsInASPNet20.aspx

[6]. http://maciej-progtech.blogspot.com/2010/01/web-parts-w-aspnet-przykady-w-c-csharp.html