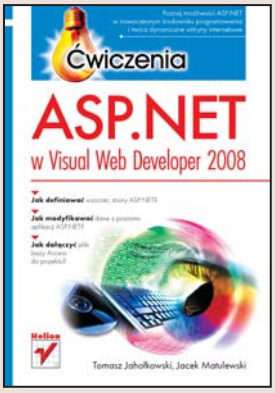
ASP.NET w Visual Web Developer 2008. Ćwiczenia



Autor: Tomasz Jahołkowski, Jacek Matulewski
ISBN: 978-83-246-1290-1
Format: B5, stron: 160

Poznaj możliwości ASP.NET w nowoczesnym środowisku programowania i twórz dynamiczne witryny internetowe

- Jak definiować wzorzec strony ASP.NET?
- Jak modyfikować dane z poziomu aplikacji ASP.NET?
- Jak dołączyć pliki bazy Access do projektu?

ASP.NET to technologia tworzenia dynamicznych stron internetowych i usług sieciowych po stronie serwera, działająca w oparciu o platformę NET. Narzędzie to doskonale sprawdza się w nowoczesnym i bezpłatnym środowisku programistycznym Visual Web Developer 2008. ASP.NET umożliwia korzystanie z dowolnego języka dostępnego na platformie .NET. Wprowadzany kod jest kompilowany, co oznacza zwiększenie wydajności i jednocześnie daje możliwość sprawdzenia jego syntaktycznej poprawności przed publikacją.

Książka „ASP.NET w Visual Web Developer 2008. Ćwiczenia" przeznaczona jest dla początkujących programistów, pragnących tworzyć witryny ASP.NET, którzy zetknęli się już z jakimkolwiek językiem programowania. Dzięki temu podręcznikowi nauczysz się korzystać z opisanych narzędzi, działających w nowoczesnym środowisku Visual Web Developer 2008. Dowiesz się, jak zaprojektować interfejs, zaprogramować silnik strony, tworzyć arkusze stylów oraz bazy danych. Na konkretnym przykładzie strony domowej nauczyciela poznasz mechanizmy autoryzacji w ASP.NET, a także sposoby dodawania i edycji danych dla poszczególnych użytkowników.

- Tworzenie interfejsu
- Programowanie silnika strony ASP.NET
- Budowanie i stosowanie arkuszy stylów
- Tworzenie bazy danych na serwerze SQL Server 2005
- Tworzenie bazy danych Access
- Technologia LINQ to SQL
- ASP.NET, ADO.NET i LINQ
- Mechanizmy autoryzacji ASP.NET
- Publikowanie aplikacji

Budowanie dynamicznych witryn internetowych jest prostsze, niż myślisz!

Wydawnictwo Helion
ul. Kościuszki 1c
44-100 Gliwice
tel. 032 230 98 63
e-mail: helion@helion.pl

Spis treści

Praktyka projektowania stron ASP.NET

Wzorzec strony (master page)

W przypadku witryny zawierającej kilka stron warto posłużyć się wzorcem. **Wzorzec** (ang. *master page*) to zwykła strona ASP.NET zapisana do pliku z rozszerzeniem *.master*. Wyróżnia ją jednak to, że zawiera komponenty `ContentPlaceHolder`, które rezerwują miejsce do wypełnienia przez strony korzystające ze wzorca. Poza tym wzorzec może zawierać zwykłe elementy HTML, jak i komponenty ASP.NET.

Wzorzec służy jako szablon pozostałych stron projektu. Tworząc nowe strony projektu, możemy wskazać ich wzorzec, a wówczas w widoku projektowania, zamiast edytować całą stronę, będziemy edytować jedynie te miejsca, które we wzorcu zostały zarezerwowane komponentami `ContentPlaceHolder`.

W najprostszym przypadku można posłużyć się wzorcem do ujednolicenia nagłówków i stopek wszystkich stron witryny — wówczas wzorzec zawiera tylko jeden komponent `ContentPlaceHolder`. I właśnie na takim przykładzie nauczymy się teraz tworzenia wzorców i korzystania z nich.

ĆWICZENIE

2.1 Przygotowywanie projektu

Dodanie wzorca do projektu i zastosowanie go w odniesieniu do istniejących stron jest możliwe, ale wymaga edycji szablonu strony. Odłożymy to zatem na później, a teraz utworzymy zupełnie nowy projekt, w którym pierwszą czynnością będzie usunięcie domyślnie utworzonej strony *Default.aspx*. Następnie zdefiniujemy wzorzec i utworzymy korzystające z niego strony.

1. Tworzymy nowy projekt:
   a) z menu *File* wybieramy *New Web Site…*,
   b) zaznaczamy pozycję *ASP.NET Web Site*,
   c) z rozwijanej listy *Location* wybieramy *File System* (wartość domyślna),
   d) a z rozwijanej listy *Language — Visual C#*,
   e) klikamy *OK*.
2. Z projektu usuwamy stronę *Default.aspx*:
   a) zaznaczamy ją w oknie projektu (podokno o nazwie *Solution Explorer*),
   b) rozwijamy menu kontekstowe,
   c) wybieramy z niego polecenie *Delete*,
   d) pojawi się pytanie o potwierdzenie, w którym klikamy przycisk *OK*.

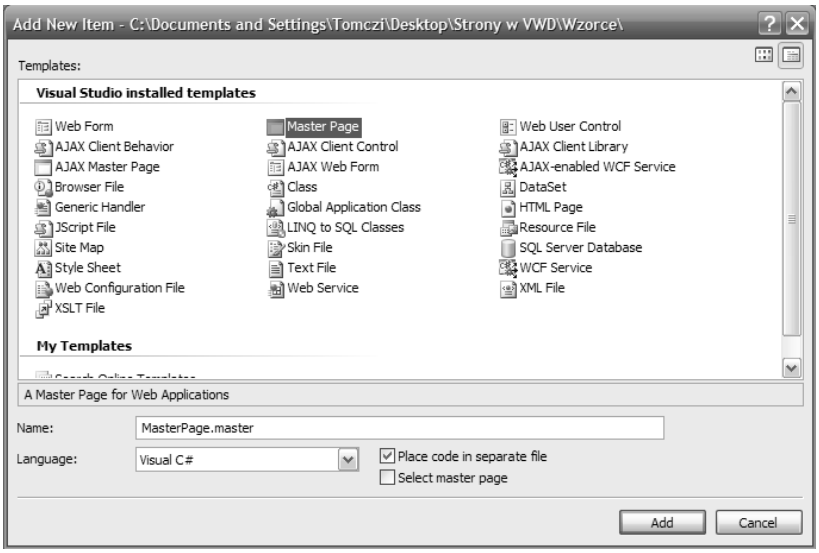
To usunie plik strony nie tylko z projektu, ale także z dysku. W tym miejscu umieścimy nową wersję strony o nazwie *Default.aspx*, ale korzystającą ze wzorca. Wcześniej musimy oczywiście przygotować wzorzec. Nie będziemy w tym zbyt wymyślni — zdefiniujemy prosty nagłówek oraz stopkę strony i zadowolimy się jednym komponentem `Content ↪PlaceHolder`.

Nie należy mylić usuwania pliku (także z dysku), a więc polecenia *Delete*, z usuwaniem pliku z projektu, tj. z poleceniem *Exclude From Project*.

ĆWICZENIE

2.2 Definiowanie wzorca

1. W oknie projektu zaznaczamy główną gałąź reprezentującą projekt całej witryny (a nie katalog *App_Data*).
2. Z menu *File* wybieramy *New File*….
3. W oknie *Add New Item* (rysunek 2.1) zaznaczamy pozycję *Master Page*.



Rysunek 2.1. *Polecenie dodawania pliku jest czułe na zaznaczoną pozycję w oknie projektu — aby zobaczyć wszystkie możliwe rodzaje plików, należy zaznaczyć pozycję odpowiadającą całemu projektowi*

4. Klikamy *Add*. Do projektu zostanie dodany nowy plik *MasterPage.master*. W edytorze zobaczymy jego kod. Zawiera on dwa komponenty `ContentPlaceHolder`: jeden w nagłówku (w znaczniku `head`), drugi w ciele strony (w znaczniku `body`). My będziemy się teraz interesować przede wszystkim tym drugim.
5. Przejdźmy do widoku projektowania nowej strony (zakładka *Design* w dole okna). Zobaczymy na niej komponent klasy `ContentPlaceHolder` — ten zdefiniowany w znaczniku `body`.

Rezerwuje on miejsce, które będzie zapełniane przez strony korzystające z naszego wzorca. Jeżeli chcemy dodać więcej komponentów `ContentPlaceHolder`, znajdziemy je na zakładce *Standard* w *Toolbox*.

6. Umieśćmy powyżej i poniżej tego komponentu jakiś tekst pełniący rolę nagłówka i stopki stron naszej witryny. Przykład widoczny jest na rysunku 2.2.

Rysunek 2.2. *Osadzanie komponentu ContentPlaceHolder we wzorcu witryny*

ĆWICZENIE

2.3 Strona korzystająca ze wzorca

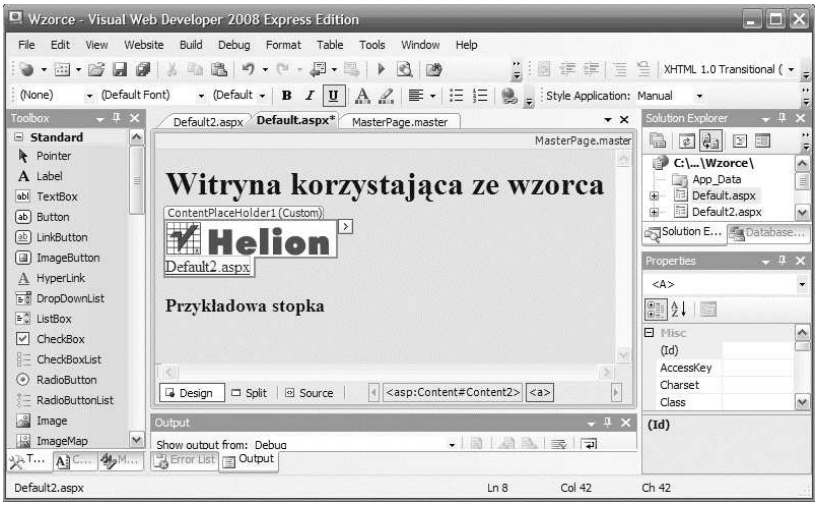
Nasz wzorzec jest prosty, żeby nie powiedzieć prymitywny, ale nie o estetykę teraz chodzi, a o ideę. Stworzymy zatem przykładowe strony, które będą z tego wzorca korzystać.

1. Z menu *File* wybieramy ponownie pozycję *New File*.
2. Tym razem zaznaczamy ikonę *Web Form*.
3. Koniecznie musimy zaznaczyć pole opcji *Select master page*. Tylko w momencie tworzenia strony można wskazać jej wzorzec.

4. Zalecam również zaznaczenie opcji *Place code in separate file*, dzięki której ewentualne metody zdarzeniowe będą umieszczone w osobnym pliku.
5. Musimy wskazać jeszcze nazwę pliku strony — domyślnie jest to *Default.aspx* — oraz język użyty do programowania metod zdarzeniowych. Jak już się pewnie Czytelnik zorientował, w tej książce zalecamy używanie C#.
6. Wreszcie klikamy *Add*.
7. Natychmiast pojawi się okno *Select a Master Page*. Wskazujemy w nim stronę *MasterPage.master* i klikamy *OK*.

Ciekawe możliwości daje zagnieżdżanie tworzonej strony wzorca w innych wzorcach. Można tego użyć między innymi do bardziej elastycznej kontroli wyglądu stron z różnych działów jednej witryny.

Po utworzeniu strony znajdziemy się w edytorze kodu. Przejdźmy niezwłocznie do widoku projektowania. Zobaczymy w nim stronę wzorca, ale poza obszarem wyznaczonym przez komponent `ContentPlaceHolder` jest ona niedostępna do edycji (rysunek 2.3). Miejscem, w którym możemy umieszczać nasze komponenty, jest wyłącznie miejsce zarezerwowane wcześniej we wzorcu.



Rysunek 2.3. *Edycja stron korzystających ze wzorca ogranicza się do obszarów wyznaczonych we wzorcu przez komponent ContentPlaceHolder*

Wypełnijmy miejsce przeznaczone na stronę jakąś przykładową zawartością. Umieśćmy w nim np. komponent `HyperLink`. Za pomocą okna *Properties* do jego właściwości `ImageUrl` przypiszmy adres *http://*
➥*helion.pl//img/logo162_35.gif*, natomiast do właściwości `NavigateUrl` adres *http://helion.pl*. Na podglądzie powinniśmy zobaczyć natychmiast logo Wydawnictwa Helion (por. rysunek 2.3). Możemy stworzyć teraz kolejne strony korzystające z tego samego wzorca, który ujednolici ich wygląd. Wzorzec poprawia zatem spójność całej witryny. Nie do przecenienia jest fakt, że stopkę i nagłówek wszystkich stron witryny kontrolujemy z jednego pliku, zatem jeżeli chcemy coś do nich dodać lub je zmienić, wystarczy edytować tylko plik wzorca.

ĆWICZENIE

2.4 Stosowanie wzorca w istniejącej stronie

Może się zdarzyć, że zechcemy dodać do rozwijanego od pewnego czasu projektu witryny wzorzec ujednolicający wygląd stron. Wówczas stajemy przed problemem użycia nowego wzorca w istniejących już stronach. Wymaga to od programisty modyfikacji kodu w pliku *.aspx*, ale jest jak najbardziej wykonalne. Załóżmy zatem, że projekt zawiera wzorzec i stronę nie korzystającą ze wzorca oraz że chcemy ową stronę w ten wzorzec wtłoczyć. Wówczas należy:

1. Przejść do pliku *.aspx* strony (widok *Source*).
2. W dyrektywie `Page` umieścić atrybut `MasterPageFile`, podając nazwę pliku wzorca.
3. Usunąć wszystkie niepotrzebne znaczniki, a pozostawić tylko te, które stanowią ciało strony (samo wnętrze znacznika `body`).
4. Sprecyzować, w którym kontenerze umieścimy ciało strony. W tym celu wszystkie znaczniki oprócz `Page` trzeba umieścić w znaczniku `asp:Content`, dla którego należy ustawić trzy atrybuty:
   a) `ID` z dowolną wartością,
   b) `ContentPlaceHolderID` — tutaj przy ustawianiu wartości automatycznie otrzymamy do wyboru wszystkie kontenery dostępne we wzorcu,
   c) `runat` z wartością `serwer`.
5. Jeżeli w kodzie strony znajduje się znacznik `form` z atrybutem `runat="server"`, należy go także usunąć.

Listing 2.1 prezentuje kod strony, która została stworzona bez wzorca, a następnie przystosowana do pokazania swojej zawartości (przycisk z etykietą) w komponencie `ContentPlaceHolder1` używanego dotąd w tym rozdziale wzorca.

Listing 2.1. *Kod przekreślony został usunięty z domyślnego kodu strony bez wzorca. Kod z szarym tłem dodano, aby strona współpracowała ze wzorcem*

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs"
➥Inherits="_Default" MasterPageFile="~/MasterPage.master" %>

<!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 runat="server">
    <title>Untitled Page</title>
</head>
<body>
<asp:Content ID="dowolna" ContentPlaceHolderID="ContentPlaceHolder1"
➥runat="server">
    <form id="form1" runat="server">
    <div>

        //przykładowa zawartość strony
        Etykieta:<br />
        <asp:Button ID="Button1" runat="server" onclick="Button1_Click"
            Text="Hello World!" />

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

Site map i komponent SiteMapPath

Pozostańmy przy tym samym projekcie i dorzućmy do niego jeszcze kilka stron. Jeżeli witryna ma więcej stron (mowa o kilkunastu, kilkudziesięciu), warto pomyśleć o *site map* — mapie witryny. W ASP.NET przygotowanie takiej mapy polega na utworzeniu pliku XML o nazwie *Web.sitemap*, w którym znajduje się zhierarchizowana grupa elementów `siteMapNode`. W atrybutach każdego z nich wskazujemy adres strony

z witryny, jej tytuł i ewentualnie opis. Struktura znaczników ma odzwierciedlać logiczną strukturę strony, wskazując strony nadrzędne i ich podstrony. Ilość stopni hierarchii jest w zasadzie dowolna.

W naszym projekcie jest tylko kilka stron (załóżmy, że trzy: *Default.aspx*, *Default2.aspx* i *Default3.aspx*), ale i my zdefiniujemy plik *Web.sitemap*. Przyjmijmy, że *Default.aspx* jest stroną tytułową jakiegoś działu witryny o nazwie „Łącza do ważnych stron", a *Default2.aspx* i *Default3.aspx* są zwykłymi stronami tego działu.

ĆWICZENIE

2.5 Tworzenie mapy witryny (site map)

1. Z menu *File* wybieramy pozycję *New File…* i w oknie *Add New Item* wskazujemy pozycję *Site Map*.
2. Powstanie plik *Web.sitemap*. Uzupełniamy go według wzoru z listingu 2.2.

Listing 2.2. *Plik Web.sitemap to plik XML opisujący logiczną strukturę witryny ASP.NET na potrzeby komponentów nawigacyjnych*

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
  <siteMapNode url="" title="Nazwa witryny"  description="Opis witryny">
    <siteMapNode url="Default.aspx" title="Linki do stron"
    ↪description="Linki do stron związanych z projektem" >
      <siteMapNode url="Default2.aspx" title="Link do ASP.NET"
      description="Linki do oficjalnej strony ASP.NET" />
      <siteMapNode url="Default3.aspx" title="Link do MSDN"
      ↪description="Link do strony dokumentacji MSDN2" />
    </siteMapNode>
  <siteMapNode url="" title=""  description="" />
  </siteMapNode>
</siteMap>
```

W elemencie `siteMap` może być tylko jeden element `siteMapNode`, więc ewentualną rozbudowę powyższej struktury należy zacząć od trzeciego poziomu zagnieżdżenia elementów XML.

Plik *Web.sitemap* może być źródłem danych dla komponentów umieszczanych na stronach, które będą pozwalać internaucie na zorientowanie się w pozycji oglądanej strony w strukturze całej witryny i umożliwią szybkie przejście do innych jej stron. Na początek przyjrzyjmy się komponentowi `SiteMapPath`.

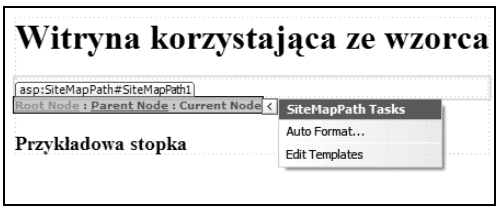
ĆWICZENIE

2.6 Informacja o pozycji bieżącej strony w strukturze witryny

1. Przejdźmy do widoku projektowania wzorca *MasterPage.master*.
2. Umieśćmy na nim komponent `SiteMapPath` z zakładki *Navigation*. Komponent ten pokazuje ścieżkę aktualnej strony w strukturze zdefiniowanej w pliku *Web.sitemap* (rysunek 2.4).

Rysunek 2.4. *Działanie komponentu SiteMapPath jest uwarunkowane obecnością pliku Web.sitemap. Zdefiniowane w nim opisy pojawiają się w okienkach podpowiedzi*

3. Zwróćmy uwagę na mały trójkącik widoczny po prawej stronie górnej krawędzi nowego komponentu, w sytuacji gdy ten jest zaznaczony. Jeżeli go klikniemy, pojawi się lista typowych zadań dotyczących tego komponentu (rysunek 2.5). W tym przypadku składa się ona z dwóch pozycji *Auto Format* oraz *Edit Templates*. Pierwsze polecenie służy do niemal automatycznego konfigurowania wyglądu komponentu. Wybierzmy np. szablon *Colorful*.



Rysunek 2.5. *Podręczna lista zadań to zbiór najczęściej wykorzystywanych kreatorów związanych z komponentem*

ĆWICZENIE

2.7 Menu strony

Innym zastosowaniem mapy witryny jest automatycznie tworzone menu i drzewo zawierające strony uwzględnione w tym pliku. Zacznijmy od menu. Dodamy je do wzorca — tym razem ponad komponentem rezerwującym miejsce dla stron.

1. Przechodzimy do widoku projektowania wzorca *MasterPage.master*.
2. Ponad komponentem `ContentPlaceHolder` umieszczamy komponent `Menu` z zakładki *Navigation*.
3. W liście podręcznej z rozwijanej listy *Choose Data Source* wybieramy *<New data source…>*.
4. Pojawi się kreator *Data Source Configuration Wizard* pozwalający na wybór źródła danych, na podstawie których zostanie utworzone menu. Może ono zostać zbudowane w oparciu o dowolny plik XML lub nasz gotowy plik *Web.sitemap*. Wybieramy oczywiście tę drugą możliwość, zaznaczając ikonę *Site Map*. Klikamy *OK*. Utworzony zostanie komponent `SiteMapDataSource1`.
5. Nam pozostaje tylko sformatować wygląd menu. Proponuję również tym razem wybrać szablon *Colorful* (rysunek 2.6).

ĆWICZENIE

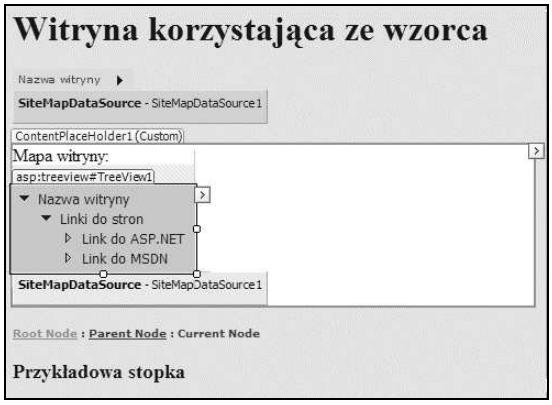
2.8 Drzewo pokazujące strukturę stron w witrynie

Na zakładce *Navigation* jest jeszcze jeden komponent, na który też warto zwrócić uwagę. Jest to drzewo `TreeView`, które prezentuje strukturę witryny. Nadaje się bardziej na osobną stronę niż do umieszczenia w nagłówku lub stopce stron.

1. Z menu *File* wybieramy *New File*.
2. W oknie *Add New Item* zaznaczamy *Web Form*, wybierając opcję *Select master page*, i zmieniamy nazwę pliku na *MapaWitryny.aspx*.
3. Klikamy *Add*. W nowym oknie wybieramy wzorzec i klikamy *OK*.
4. Przechodzimy do widoku projektowania nowej strony.
5. Na dostępnym obszarze umieszczamy komponent `TreeView` z zakładki *Navigation*.

Rysunek 2.6. *Użycie tego samego stylu do formatowania menu i ścieżki pokazującej pozycję w strukturze witryny to zalążek estetycznego i spójnego wizerunku wszystkich stron witryny*

6. Postępując identycznie jak w przypadku menu, tworzymy źródło danych korzystające z mapy witryny (niestety, nie można użyć gotowego komponentu `SiteMapDataSource1` widocznego w obszarze wzorca).
7. Formatujemy drzewo, wybierając z podręcznej listy zadań pozycję *Auto Format…*. Proponuję użyć stylu *Arrows 2* (rysunek 2.7).



Rysunek 2.7. *Automatycznie generowana mapa witryny w widoku projektu*

8. Nową stronę warto dopisać do mapy witryny, tworząc węzeł równorzędny z *Default.aspx*.

Poza zwykłą nawigacją, jaką umożliwia komponent `TreeView`, można definiować metody zdarzeniowe związane z kliknięciem różnych pozycji drzewa (zdarzenie `SelectedNodeChanged`).

Dwa słowa na temat kaskadowych arkuszy stylów

Wiemy już, że w projektach ASP.NET możemy w znacznym stopniu, w zasadzie nawet całkowicie, odseparować kod C# od szablonu HTML strony. W ten sposób oddzielony zostaje kod odpowiedzialny za statyczny wygląd stron witryny od metod kształtujących ich dynamikę. Do tych dwóch etapów projektowania dochodzi trzeci, w którym za pomocą kaskadowych arkuszy stylów wpływamy na estetykę witryny. Podobnie jak w przypadku kodu C#, także arkusze stylu mogą być całkowicie odseparowane w plikach *.css*, a przez to ich rozwój, podobnie jak kodu C#, może być z łatwością powierzony innym osobom niż te, które rozwijają kod z plików *.aspx*.

W kilku poniższych przykładach przedstawię podstawowe narzędzia służące do budowania kaskadowych arkuszy stylów.

ĆWICZENIE

2.9 Tworzenie arkuszy stylów

Kaskadowe arkusze stylów (ang. *cascade style sheet*) to kolejne obok wzorca narzędzie ujednolicenia stron witryny, a jednocześnie zcentralizowania kontroli nad ich wyglądem. I w tym przypadku wsparcie ze strony VWD jest godne pochwały.

1. Z menu *File* wybieramy *New File*….
2. W oknie *Add New Item* zaznaczamy pozycję *Style Sheet* i jeżeli odczuwamy taką potrzebę, zmieniamy nazwę pliku; potwierdzamy kliknięciem *OK*.

W edytorze zobaczymy niemal pusty plik, który zawiera jedynie tekst:

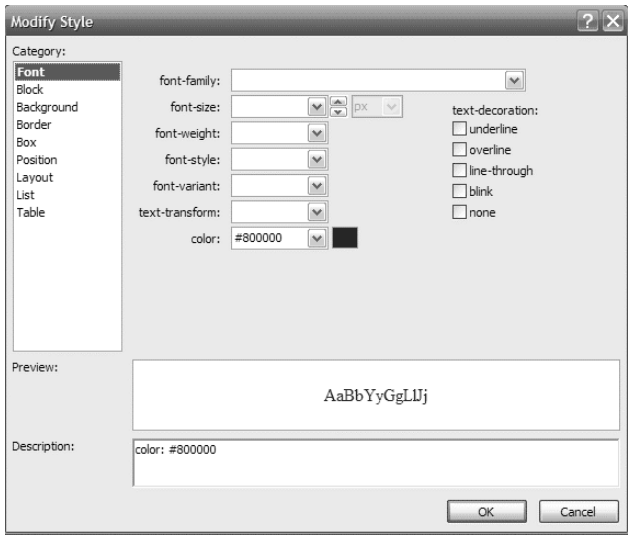
```
body {
}
```

To zalążek stylu związanego ze zwykłym tekstem umieszczonym na stronie (tekst między znacznikami BODY w kodzie HTML). Na szczęście nie musimy znać się na formacie arkuszy stylów, bo VWD zawiera proste narzędzie pozwalające na ich definiowanie. Zacznijmy od rozbudowania reguły formatowania dotyczącej prostego tekstu.

ĆWICZENIE

2.10 Formatowanie tekstu na stronie

1. W edytorze ustawiamy kursor (edycji, nie myszy) między nawiasami istniejącej reguły stylu.
2. Klikamy na pasku narzędzi przycisk *Build Style…* (dostępny jest również z menu kontekstowego).
3. W oknie *Modify Style* (rysunek 2.8) możemy wybrać format i kolor czcionki, tła, list i innych elementów umieszczonych na stronie. My ograniczymy się do sformatowania czcionki, dlatego klikamy pozycję *Font* na liście zakładek widocznej z lewej strony okna.



Rysunek 2.8. *Definiowanie reguły stylu dla znacznika body*

4. Rozwijamy listę zatytułowaną *color* i klikamy *More Colors…*:
   a) w oknie *More Colors* odnajdujemy np. kolor *maroon* (jeden z brązowych), który będzie dobrze współgrał ze stylem formatowania wybranym w menu i w innych komponentach nawigacyjnych; dla większej elastyczności kolory reprezentowane są przez liczby szesnastkowe[1];
   b) klikamy *OK*, aby zamknąć kreator stylu.
5. Następnie, korzystając z ikony *Add Style Rule* na pasku narzędzi (lub analogicznego polecenia menu kontekstowego), tworzymy nowy styl dla znacznika `A` (tj. dla umieszczonych na stronie łączy):
   a) ponownie uruchamiamy kreator stylów (przycisk *Build Style…* na pasku narzędzi) i zakładkę *Font*;
   b) tym razem wybieramy kolor *Black*;
   c) w części *text-decoration* zaznaczamy pole *None*.
6. Po tym zdefiniujmy jeszcze jedną regułę formatowania dla `A:hover` (łącze po naprowadzeniu na niego kursora), w którym kolor ustalamy na *Red*, a w części *text-decoration* (por. rysunek 2.8) włączamy opcję *Underline*.

Po tych czynnościach plik kaskadowego arkusza stylów (plik z rozszerzeniem *.css*) powinien wyglądać jak na listingu 2.3.

Listing 2.3. *Zawartość pliku kaskadowych arkuszy stylów*

```
body
{
    color: #800000;
}
A
{
    color: #000000;
    text-decoration: none;
}
A:hover
{
    color: #FF0000;
    text-decoration: underline;
}
```

[1] Bardziej egzotyczne nazwy kolorów mogą być niezrozumiałe dla starszych przeglądarek.

ĆWICZENIE

2.11 Stosowanie arkuszy stylów

Czas, aby arkusz wykorzystać do formatowania stron naszej przykładowej witryny:

1. Przejdźmy na stronę *Default.aspx*.
2. W widoku projektowania dodajmy do niej prosty tekst (wpisując go w polu `Content`) oraz komponent `HyperLink` ze skonfigurowaną właściwością `NavigateUrl` i etykietą (właściwość `Text`).
3. Przejdźmy do widoku projektowania pliku wzorca *MastepPage.master*.
4. Przeciągnijmy z okna projektu (*Solution Explorer*) utworzony plik *.css*. Do kodu strony dodany zostanie element `<link href="StyleSheet.css" rel="stylesheet" type="text/css" />`, dzięki któremu wzorzec i wszystkie używające go strony będą korzystały z arkusza i zdefiniowanych w nim stylów.

W podglądzie wzorca i podglądzie stron, które z niego korzystają, zobaczymy zmianę — tekst zmieni kolor na brązowy, łącza na jasnobrązowy. Ponadto łącza pozbawione zostały podkreślenia. Jeżeli obejrzymy stronę w przeglądarce, to zobaczymy, że kolor łączy zmienia się na pomarańczowy, jeżeli naprowadzić na nie kursor myszy, oraz że pojawia się wówczas pod nimi podkreślenie.

Jeżeli witryna nie ma wzorca, arkusz należy dodać do każdej strony osobno. To samo dotyczy stron w naszej witrynie, które nie korzystają ze wzorca.

Można również edytować indywidualny styl poszczególnych komponentów na stronach. W ich menu kontekstowym znajduje się pozycja *Style…*, która uruchamia okno *Style Builder*, widoczne na rysunku 2.8. Jeżeli natomiast chcemy użyć istniejących klas stylu, w oknie *Properties* odnajdujemy właściwość `CssClass` i tam wpisujemy nazwę klasy zdefiniowanej w arkuszu stylu. Klasy można definiować w pliku *.css*, dodając regułę i zaznaczając opcję *Class name*.

Czas życia sesji i aplikacji ASP.NET

Po pierwszym wywołaniu witryny na serwerze WWW uruchamiana jest odpowiedzialna za nią aplikacja ASP.NET i od tego momentu najczęściej nie przestaje działać aż do zakończenia pracy serwera. Jednocześnie „w obrębie aplikacji" powstaje sesja, która związana jest z tym pierwszym żądaniem. Kolejne żądania od innych przeglądarek-klientów powodują tworzenie następnych sesji. Po opuszczeniu przez internautę witryny sesje kończą pracę, ale aplikacja wciąż jest na posterunku. To zasadnicza zmiana w porównaniu do starszych technologii rozszerzeń serwerów WWW, w których aplikacja uruchamiana była tylko po to, aby przetworzyć otrzymane od przeglądarki żądanie i wygenerować nowy kod HTML. W ASP.NET mamy do czynienia ze stale pracującą aplikacją, która przechowuje swój stan. W tym jest podobna do technologii ***Java Server Pages*** firmy Sun. W szczególności ciągłość pracy aplikacji powoduje, że możemy zapamiętać jakieś dane na jednej stronie witryny, a wykorzystać na innej. Do tego służy zbiór danych (zmiennych) sesji. Nie ma także problemu, aby utworzyć zmienną przechowywaną nawet po zamknięciu sesji. Wystarczy umieścić ją w zbiorze danych aplikacji. A co w przypadku zamknięcia serwera WWW? Oczywiście oba zbiory zostaną utracone. Jednak dane mogą być przechowywane w plikach. Możemy do tego wykorzystać stare dobre *cookies*, szczególnie jeżeli dane dotyczą konkretnego użytkownika — w tym przypadku dane przechowywane są po stronie klienta; można również wykorzystać pliki XML lub wręcz jedną z baz danych ADO.NET. To już oczywiście po stronie serwera. Mamy tu pełną swobodę działania.

Aby lepiej prześledzić cykl życia aplikacji ASP.NET i jej sesji, stwórzmy metody zdarzeniowe związane z kluczowymi momentami życia aplikacji i sesji. W tym celu do aplikacji, którą stworzyliśmy w poprzednim podrozdziale, dodamy specjalny plik *Global.asax*.

ĆWICZENIE

2.12 Tworzenie pliku Global.asax

1. Z menu *File* wybieramy *New File*.
2. W dobrze nam już znanym oknie zaznaczamy pozycję *Global Application Class*.